1984

Implementation of a general finite element code on an IBM PC comppatible microcomputer.

Ruesch, Rehe E.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

IMPLEMENTATION OF A GENERAL FINITE ELEMENT CODE ON AN IBM PC COMPATIBLE MICROCOMPUTER

by

Rehe E. Ruesch

September 1984

Thesis Advisor: G. Cantin

Approved for public release; distribution unlimited.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Implementation of a General Finite Element Code on an IBM PC Compatible Microcomputer		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; September 1984
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Rehe E. Ruesch		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		12. REPORT DATE September 1984
		13. NUMBER OF PAGES 280
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Finite Elements
Structural Analysis
IBM-PC
Columbia MPC
Microcomputers
Microcomputer Applications
16 Bit Microcomputers
Microsoft FORTRAN 77
Computer Programs

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The practicality of using microcomputers to solve systems of equations of several hundred unknowns has been demonstrated. However, machine and software limitations of eight bit processors made the construction of useful finite element programs very difficult, and severely limited the size of problems which could be solved in a reasonable amount of time. The introduction of the sixteen

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

bit microprocessor has completely revolutionized the microcomputer industry, and many of the limitations of the eight bit systems have been eliminated. The new microcomputers have made mainframe-like computing power available to everyone, at a very reasonable cost. For many reasons, however, there are few general finite element programs available for the microcomputer today. A general finite element program of moderate complexity called MEF ("Méthode des Eléments Finis") is adapted for implementation on the IBM PC-XT and the COLUMBIA MPC microcomputers. The resulting implementation is verified, and results are compared with other finite element systems.

Implementation of a General Finite Element Code on an IBM PC Compatible Microcomputer

by

Rehe E. Ruesch
Lieutenant Commander, United States Navy
B.S., Purdue University, 1976

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1984

ABSTRACT

The practicality of using microcomputers to solve systems of equations of several hundred unknowns has been demonstrated. However, machine and software limitations of eight bit processors made the construction of useful finite element programs very difficult, and severely limited the size of problems which could be solved in a reasonable amount of time. The introduction of the sixteen bit microprocessor has completely revolutionized the microcomputer industry, and many of the limitations of the eight bit systems have been eliminated. The new microcomputers have made mainframe-like computing power available to everyone, at a very reasonable cost. For many reasons, however, there are few general finite element programs available for the microcomputer today. A general finite element program of moderate complexity called MEF ("Méthode des Eléments Finis") is adapted for implementation on the IBM PC-XT and the COLUMBIA MPC microcomputers. The resulting implementation is verified, and results are

TABLE OF CONTENTS

LIST OF TABLES

I. INTRODUCTION

A. BACKGROUND

The microcomputer was introduced in the marketplace a little over a decade ago, and has proceeded to develop at an astounding rate. The very large-scale integrated (VLSI) microcomputer of today performs at speeds four to six orders of magnitude greater than first generation computers. The power, utility and versatility of these microcomputers is such that their capability exceeds that of second generation computers, and they are as powerful as many minicomputer systems in a variety of applications. Indeed, the increasing capabilities of microcomputers have driven the latest, high-speed minicomputers to attain the versatility and performance standards of the former medium-scale mainframe computers [Ref. 1: pp. iii-8].

Today, an unprecedented amount of computing power is available at a price which can be afforded by even the smallest of engineering firms and research groups. Two years ago, research into the implementation of finite element software on eight bit microcomputers was conducted by Mulholland [Ref. 2]. At that time, the cost of the computer system used for his research was approximately $6,000 [Ref. 2: p. 15]. The cost of each of the systems used in this research is about the same, but there is

hardly a comparison in capability between the Apple II Plus used by Mulholland, and the IBM PC-XT of today. The problem today is that there is not a large library of engineering software available for engineering firms and research groups to take advantage of. Wilson [Ref. 6] states that less than one percent of current day finite element analysis is conducted on microcomputers. This lack of software leaves small firms and research groups (who do not have the staff, resources and time to develop extensive programs themselves) unable to take advantage of the computing power available to them. For this reason, there is a need to develop engineering software which will take advantage of the microcomputer's capabilities. Unfortunately, at this time, the limitations and capabilities of the software/microcomputer combination are largely unexplored. This thesis will attempt to shed light on the capabilities of the sixteen bit microcomputer to perform general finite element analysis.

1. Eight Bit Micros and Finite Elements

When first introduced, microcomputers used an eight bit architecture which provided several stumbling blocks to the implementation of engineering software in general. The most significant of these stumbling blocks was the memory size limitation, and the second was a limited instruction set. The maximum addressable memory of the most advanced of these systems was 65,536 bytes. This address space was

often limited, even further, by the presence of read only memory (ROM) chips which contained significant portions of the operating system for the computer. The result was a severe limitation to the size of application software, as well as the size of data objects. The instruction set limitation was significant because it complicated the implementation of high level language compilers, and almost all engineering applications are dependent on the availability of high level languages. These two problems combined to cause another problem which was the immaturity of support software. The limited instruction sets required more code to implement desired features, yet the small memory size restricted the amount of code severely. The result was that operating systems, compilers, and interpreters were notorious for the things they could not do. Nevertheless, there have been a number of commercial as well as academic implementations of finite element codes on eight bit microcomputers. All of these implementations are limited to relatively small problems, and almost all are special application programs which solved only one type of problem (typically beams, trusses, or frames).

As time progressed, the hardware and operating systems of microcomputers matured. With the advent of high speed floppy diskette drives and disk operating systems, the idea of using out-of-core linear equation solvers to solve larger systems of equations on microcomputers became

achievable. Mulholland [Ref. 2: pp. 36-46] demonstrated the ability of eight bit machines to produce an acceptable result using an out-of-core technique. As might be expected, solution times were somewhat slow, but the method increased the size of problem which could be solved with a limited amount of memory. After verifying the utility of the eight bit machine and out-of-core solver combination, Mulholland [Ref. 2: pp. 52-70] continued his investigation by implementing a modification of Mallory's [Ref. 3] STAP-NPS on the Apple-II Plus microcomputer.

The result was a finite element system which was cumbersome to use, and supported only one element type. The system required the attention of the user to shift five floppy diskettes between four disk drives in response to requests from the run time system. In his tests, over two hours were required to solve a system of 160 equations having a half-bandwidth of 64. Mulholland's conclusion [Ref. 2: pp. 72-74] was that the system he used was not a suitable tool for serious finite element work. He cited six primary reasons why the system was inadequate for the application, but it is significant to note that five of the six reasons were actually operating system/compiler limitations. In other words, five of the six were due to immaturity of support software for the Apple-II Plus system at the time.

2. Sixteen Bit Microcomputer Introduced

In 1981 and 1982 the sixteen bit LSI microcomputer was introduced to the market. While the speed of these machines was as much as four or five times greater than the eight bit predecessors, the largest advantages were realized by the improved architecture and instruction sets. Rao [Ref. 1: p. 205] cites a ten fold improvement in execution time for the Intel 8086 over the Intel 8080A while the increase in clock speed was, at most, 4 times that of the 8080A. Obviously, the influence of the architecture and instruction set is strongly significant. One of the most important improvements delivered by the sixteen bit processors was the amount of addressable memory; the smallest address space among the various architectures was 1024 kilobytes. This is not meant to imply that application software was able to take advantage of that address space. There were no compilers, at the time, that would allow addressing outside a 64 kilobyte page. Even today, compilers that allow addressing beyond the 64 kilobyte page are just beginning to enter the market. However, a tremendous amount of support software such as compilers, interpreters, spread sheets, etc. were no longer limited to 64 kilobytes of memory. As a result, support software began to grow in size and capability. In addition to the increase in address space a number of the systems were able to make use of a separate coprocessor, the Intel 8087, for

numeric data processing. Although, no specific standard has been developed for comparing processing times with and without the coprocessor, most authors agree that addition of the coprocessor has been shown to increase the speed of numeric computation significantly [Ref. 4, 5].

The introduction of the sixteen bit machines, however, did not cause an immediate surge in finite element applications on microcomputers. Wilson [Ref. 6] points out that the development of engineering software is dependent upon the availability of a stable operating system, and a compatible FORTRAN compiler. While today's microcomputers are seldom marketed without an operating system, the initial versions of operating systems have been notoriously unreliable and unsophisticated. Therefore, a lag exists between the introduction of the hardware, and the development of a stable operating system and a compatible FORTRAN compiler. The lag in the case of the eight bit machines was nearly ten years, but because of the experience gained in the development of these systems the lag was shortened considerably for sixteen bit microcomputers.

At the outset of this investigation, the market boasted a variety of disk operating systems which supported floppy disks as well as the newer high-speed hard disks. There were two FORTRAN compilers available which had undergone a number of modifications and promised the maturity necessary to support finite element applications. There

was also a wide variety of peripheral plotting devices and other equipment and software to support engineering applications.

B. PURPOSE AND SCOPE OF THE INVESTIGATION

This investigation was conducted as an attempt to implement a general purpose finite element program on a sixteen bit microcomputer, with the intent of determining whether or not the resulting system was practically useful. Wilson [Ref. 6] makes the assertion that new finite element work will be done in FORTRAN, primarily, because all general purpose finite element programs, to date, have been written in FORTRAN. This author supports the assertion with the observation that FORTRAN is also the most widely used and supported language in the engineering community. In addition, FORTRAN 77 eliminates most of the practical objections to FORTRAN as a programming language. Much previous work has gone into the implementation of the more notable general, finite element programs in use today, and the construction of these programs is a project which requires considerable investment in terms of manpower and dollars [Ref. 6]. Since the purpose of the investigation was to evaluate the usefulness of the resultant program/machine combination, it was desirable to implement a system of moderate complexity in order to provide a rigorous test. Therefore, the decision was made to convert an appropriate, existing

program rather than to reinvent the wheel. The program which was chosen, called MEF ("Méthode des Eléments Finis"), was written at the Université de Technologie, Compiègne, France. Justification for the choice of MEF is provided in section 1.5 below. Initially, it was hoped that graphics, and user friendly input routines could be added to the implementation, however, time constraints limited the investigation to conversion of the existing software.

C. CHOICE OF THE MACHINE

The major considerations influencing the choice of the microcomputer for this study were:

(1) Availability and support of system hardware (including peripherals).

(2) The existence of a FORTRAN compiler compatible with the system.

(3) The existence of a compatible, FORTRAN callable, graphics package for future implementation of graphics.

(4) System cost.

(5) The existence of a wide range of technical support for hardware maintenance.

(6) The availability of a wide range of commercial software for the system.

The last two considerations are to insure that the chosen system was maintainable, and versatile. Presumably, any firm or research group considering the purchase of a

microcomputer would desire to use it for more than finite element analysis. The existence of technical support and commercially available software would be significant considerations in the choice of a system. System hardware considerations included floppy disks, hard disks, modems, printers, plotters, graphical input devices (digitizers, joy sticks, mouse, etc.), and the ability to support a large amount of memory.

At the time equipment for the project had to be chosen IBM's complete domination of the microcomputer market made the choice of the IBM PC or a PC compatible microcomputer the logical choice of hardware for the system. In addition, the IBM PC was widely available at the Naval Postgraduate School. IBM's domination of the market also spawned a tremendous industry aimed at producing peripherals and software for the IBM PC. Therefore, it was clear that capable language compilers, graphics devices, graphics software, and other software would develop more quickly and predictably for the IBM PC than for other systems.

In the end, two systems were used to conduct the investigation: an IBM PC-XT available at the Naval Postgraduate School, and a Columbia MPC. The Columbia MPC was chosen for use at home because of its lower cost and high degree of compatibility with the IBM PC.

1. Configuration of the System

Differences between the two machines chosen for this study are minimal. The description which follows is applicable to both systems with exceptions as noted.

Table I. Configuration of the System

COMPONENT	DESCRIPTION	COMMENTS
CPU	INTEL 8088 with the INTEL 8087 coprocessor	
MEMORY	512 kilobytes	
DISPLAY	color monitor with graphics adapter and a monochrome display	the Columbia supported a graphics capable monochrome monitor with a graphics adapter
MASS STORAGE	one 5.25 inch, double density, dual sided floppy diskette drive and one ten megabyte hard disk drive	the Columbia system initially supported two 5.25 inch, double density, dual sided floppy diskette drives and no hard disk drive
PRINTER	graphics capable, dot matrix, parallel printer	
SERIAL PORTS	two (one used for mainframe communications, and one for a graphics input device in support of future graphics development)	

The hard disk was not required for the development of the MEF system, however, the availability of the hard disk cut compile and linking time almost in half over the Columbia

floppy disk system. When compile and link times approached two hours on floppy disks for the complete MEF system, the increased speed of the hard disk was significant. Later in the investigation, a fifteen megabyte hard disk was added to the Columbia MPC.

At the beginning of the investigation it was impossible to determine the amount of memory which would be required to implement MEF. However, the original implementation of MEF on a VAX 780 minicomputer contained a working array consisting of 160 kilobytes. The memory size of 512 kilobytes was chosen because it was the amount which would fully populate the memory expansion board chosen, and it was felt that it would be large enough to minimize the difficulty in implementing program overlays if overlays became necessary.

Both systems run using functionally identical operating systems (MS DOS for the Columbia and PC DOS for the IBM). Indeed, no modifications of any kind were required to carry the software between the two systems. Even the compiled and linked programs could be carried between the systems.

2. CPU Speed Tests

CPU speed for the two systems is advertised to be 4.77 MHz. However, the hardware and operating systems of both machines are extensively interrupt driven. No criticism is intended of this extremely powerful method of implemention, however, it was known that the effective speed of the

system would be somewhat lower than the clock speed because of the system overhead created by extensive use of interrupts and interrupt handlers. For this reason, a simple test was devised to determine the processor speed available to a user program (apparent speed) to compare with eight bit processor speeds.

The test involved writing a simple assembly language routine which would place the processor in a loop of specified length. The system clock was accessed just before and just after the loop to compute the time spent in the loop. [Ref. 7] was used to determine the number of clock cycles in the loop, and the number of times through the loop was chosen to be large enough so that the computation time involved in accessing the clock would not significantly influence the resultant computation. The program and calculations used are detailed in Appendix A. The result of the test indicated that the apparent speed of the systems is approximately 3.56 MHz (worst case). This test was somewhat subjective, however, the result gave some insight into the minimum performance which could be expected from the system. While the 3.5 MHz speed is significantly lower than 4.77 MHz, it is still about one and a half times greater than typical eight bit processor speeds; coupled with a better architecture and instruction set it was clear that significant things could be expected.

3. Matrix Solution Demonstration

In comparison with the matrix solution tests conducted by Mulholland [Ref. 2: pp. 45-48] tests of the IBM PC-XT, and the Columbia MPC configured with 512 kilobytes of RAM yielded significantly faster solution times for even larger matrices than those tested by Mulholland. The tests were conducted using the FORTRAN program in Appendix B to solve double precision, fully populated, matrices. The solver uses an LU decomposition followed by back substitution, and takes no advantage of symmetry or bandwidth. The results are as follows:

Table II. Matrix Solution Benchmark Tests

DEGREES FREEDOM	PREDICTED SOLUTION TIME	ACTUAL SOLUTION TIME
25	unknown	0.0577 min.
32	.12 min.	.119 min.
100	3.69 min.	3.48 min.
200	29.54 min.	27.56 min.

The first test was conducted on a matrix whose storage requirements would not exceed 64 kilobytes, but would be large enough to provide a usable benchmark. The 32 degree of freedom (DOF) matrix was run for direct comparison with the results obtained by Mulholland [Ref. 2: pp. 45-46]. As can be seen from the table, the time

required for solution was small as compared with the Apple II Plus best time of 2.68 minutes, and the HP 9845 best time of 0.87 minutes. In comparison with Mulholland's reported solution time of approximately 2.5 hours for a 160 DOF problem having a bandwidth of 64, test four took only 27.56 minutes.

The solution time for the type of solver used varies with the cube of the number of degrees of freedom (i.e., a two fold increase in DOF would predict an increase in solution time of eight). The predicted times for the last three tests were based on the actual execution time for the first test. Since the actual times are even faster than the predicted times the conclusion can be drawn that the overhead of addressing outside a 64 kilobyte memory page is not excessive for this compiler, and does not seem to vary with how far outside the memory page the addressing goes. If the results of test three are used to predict test four the resultant prediction would be 27.84 minutes. As the table shows, this is very close to the actual execution time. It is important to note that for the size matrices tested, and the amount of memory available the solutions were achieved in-core. Therefore, in many respects, the comparison with Mulholland's data is informative, but not exactly fair.

The significance of the comparison is that it shows the size of problem which can be solved in a relatively

short period. The in-core solution of a 200 DOF, double precision system is important both because it has only recently become possible, and because it represents the solution of a moderate sized finite element system with a maximum bandwidth. Larger systems could be solved in-core if bandwidth were minimized, and the solution technique took advantage of symmetry and bandwidth characteristics, as is done with most modern finite element programs. For systems which take advantage of these characteristics, the solution varies as the product of DOF and the square of the bandwidth. This means that a finite element system having a bandwidth of eighty and five hundred degrees of freedom would be solved in about one third the time as test four above (after assembly), or if a factor of three is assumed for all the processing, it would take approximately the same time as test four. Even larger systems could be solved using out-of-core techniques, but a fairly large number of problems can now be solved in-core.

D. CHOICE OF COMPILER

At the start of this investigation, the Microsoft FORTRAN compiler (version 3.1) was the only one which offered features that might be used to implement arrays requiring more than 65,536 bytes of storage. In order to avoid typical limits on data sotrage, Microsoft reserved a full 64k segment for each named common block. In support of portability, the compiler offered a complete ANSI FORTRAN

77 Subset with a few extensions to the full language. Version 3.1 was used for most of the preliminary work with MEF, however, difficulties involved in restructuring the program to take advantage of the Microsoft named common block features prevented the implementation of MEF with full sized arrays. Later in the investigation, Microsoft FORTRAN was updated to version 3.2. The new version supported unrestricted array sizes, overlay support, and an improved support for the Intel 8087 math coprocessor.

E. CHOICE OF PROGRAM FOR CONVERSION

To provide a sufficiently rigorous test of the computer and software combination it was desirable to implement a general finite element program of moderate to robust complexity. The characteristics of such a program include:

1. The ability to solve a variety of problems including problems of elasticity, heat transfer, and fluid mechanics.

2. A choice of element types, and the ability to add elements as needed.

3. The ability to solve both static and dynamic problems of large size involving more than one element type. Including problems having different degrees of freedom at each node, symmetric or nonsymmetric element matrices.

4. The solution of eigen value problems.

5. The ability to solve nonlinear problems.

For the purpose of implementation of a finite element code on a microcomputer it would also be useful if the program included an out-of-core equation solver to eliminate severe restrictions on problem size, and block structured code to minimize the difficulty of developing overlays if necessary.

Practical considerations included the availability of source code and thorough documentation for the program. It was also necessary, because of time constraints, to have the source on some type of machine readable media.

One program which satisfied most of the above characteristics was called MEF. In addition, there was a version of MEF available at the Naval Postgraduate School which ran on the VAX 780 under the VMS operating system. The documentation for MEF was contained in a book [Ref. 8] which was translated from French by Professor Gilles Cantin of the Naval Postgraduate School. Therefore, this investigation chose MEF, the "Méthode des Eléments Finis," to convert for implementation on the IBM PC-XT and Columbia MPC microcomputers.

II. CONVERSION OF MEF

A. GENERAL

The version of MEF which was available at the Naval Postgraduate School was written in FORTRAN IV. The FORTRAN 77 compiler on the VAX 780 at the Naval Postgraduate School was able to compile the FORTRAN IV code with no alterations, and the result was tested using problems for which the solutions have been published [Ref. 8: pp. 447-503]. However, the FORTRAN 77 implementation on the VAX is a robust version of ANSI standard FORTRAN 77, with many extensions which provide compatibility with earlier versions of FORTRAN. The Microsoft FORTRAN version 3.1 which was used at the beginning of this study met the requirements of the ANSI subset of FORTRAN 77 with a few extensions to the full language.

In general, most of the problems that occur in converting engineering applications programs from FORTRAN IV to FORTRAN 77 are tied to the character and string handling differences between the two versions. Occasionally, problems arise in the conversion of do loops because FORTRAN IV did not prevent the poor programming practice of jumping into the range of a do loop. Additional problems arise when using the subset language because it does not support BLOCK DATA modules and requires all occurrences

of a named common block to be exactly the same length. These two problems are mentioned because of their common use in engineering applications, and because of their extensive use in MEF. Aside from these general areas, considerations specific to implementation on a microcomputer include restrictions on array size imposed by the compiler, and memory size limitations imposed by the hardware/ operating system (the operating system for the IBM PC does not support virtual memory). Version 3.1 of the Microsoft Compiler supported a maximum array size of 65,366 bytes (approximately 8,170 double precision words), and a maximum subscript value of 32,767 [Ref. 9: p. 63]. As mentioned previously, version 3.1 provided some relief with regard to memory restrictions by reserving 64 kilobytes of storage for each named common block [Ref. 10: p. 54].

The conversion of MEF took place in two major phases. The first phase used Microsoft FORTRAN version 3.1, and was unsuccessful. The second phase used Microsoft FORTRAN version 3.2 which was actually two full releases improved over version 3.1. The second phase was successful (with some restrictions) after a number of problems with the compiler (bugs) were identified and circumvented.

B. PHASE ONE

The original version of MEF was approximately 4800 lines long which included the comments (written in French).

There are very few microcomputer based text editors which will handle files of that length, and the ones that do become totally bogged down with the overhead of managing the file. It was determined that the most effective way to proceed would be compile approximately 500 lines of code on the microcomputer to identify the specific items which could be changed globally in the original code. After the necessary modifications were determined, they were made to the entire file using the VAX text editor, EDT. When the modifications were complete the large file was broken into five separate segments of approximately 1000 lines each and transferred, via modem, to five separate floppy diskettes. Five diskettes were required so there was room to edit and compile the separate modules. The Microsoft compiler creates intermediate (scratch) files which are almost twice the size of the source file. The easiest, safest place to place the scratch files is on the same diskette as the source file, and 1000 lines of source code, on the average, create scratch files which nearly fill the diskette.

FORTRAN does not support true dynamic memory allocation, and will not allow the direct manipulation of array sizes during execution. Therefore, MEF, like many other finite element program, implements a pseudo-dynamic memory allocation so that array sizes may be altered during execution. In order to do this, all arrays are declared as one dimensional arrays and stored sequentially in a single large

working array. A table of pointers is made to keep track of the beginning of each array, and as tables are deleted the separate tables are moved to accommodate the change. MEF defines the large working array, called VA, to be in blank common, and makes extensive use of named common for all other common block applications. For this reason, the basic structure of MEF had to be altered to implement its conversion with the Microsoft Compiler version 3.1. There were some 15 named common blocks in MEF, and since the compiler reserved 64 kilobytes for each one, approximately 960 kilobytes would have been used just for common block allocation. Needless to say, there was not enough memory available even if the compiler/linker combination were capable of handling the problem.

The program structural change attempted, was to switch all of the elements in named common blocks to blank common, and the single array VA from blank common to named common. It was envisioned that several named common blocks could eventually be used so that the actual size of the working array could be larger than 35,366 bytes. Considerable time was spent in effecting this change, and trying to get the resultant version of MEF to work. However, the changes were too comprehensive, and the attempt was aborted when a new version of the compiler was received in May of 1984. The improvements in version 3.2 allowed the complete abandonment of the restructuring approach, and while phase one

was unsuccessful, the time spent was not wasted. It allowed familiarization with the structure of MEF, and with the specific areas where the FORTRAN IV code needed to be altered for compatibility with subset FORTRAN 77. In addition, it provided considerable familiarization with the operating system, editors, and hardware of both the VAX and the IBM PC-XT/COLUMBIA MPC.

C. PHASE TWO

A new version of MEF which contained english comments was received about the same time as the new version of the Microsoft Compiler. Once again, proceeding as before, a smaller segment of code was used to determine the global changes required on the main body of the code, before transferring it to floppy diskettes. Because of the changes in Microsoft FORTRAN version 3.2, there were significantly fewer alterations required to the global structure of MEF; all of the initial changes pertained, strictly, to the treatment of characters and strings, and to the task of making the named COMMON blocks the same length in each reference.

1. Microsoft Version 3.2 Improvements

Improvements in the compiler which affected the implementation of MEF are as follows:

1. Support for the BLOCK DATA statement.

2. Support for arrays and COMMON blocks longer than 64 kilobytes.

3. Inclusion of A simple overlay linker (overlays were unnecessary in the end, but at the time the investigation started there was no way to tell whether or not they would be required).

4. Better support for the Intel 8087 coprocessor including implementation of the IEEE floating point math standard (the default for this version).

From the point of view of this investigator, the single most important change in the Microsoft Compiler was the support for large arrays and common blocks. This is the change which eliminated the unusual implementation of named COMMON present in version 3.1. It must be noted here, that though the common block problem was solved, the ability to address more than 64 kilobytes beyond the beginning of an array or common block was often defeated by compiler/linker bugs. Simple applications such as the array solver shown in APPENDIX B had no difficulty in compiling, linking, and producing results using arrays limited only by the amount of memory available. However, more complicated programs with numerous common blocks and arrays provide a serious challenge to the compiler/linker combination, and the results are not always gratifying. The improvement over earlier versions, however, are monumental and conversations with Microsoft Technical Support indicate that future releases of the compiler will solve the types of problems encountered in this research.

2. Small Memory Model Implementation of MEF

With the improvements mentioned above it took only a few weeks to provide a working version of MEF which was called the "Small Memory Model" (SMM). That is, no arrays were declared to be large (the working array size was cut to 2000 words). For details of the compiler structure, the reader is referred to [Ref. 12: pp. 99-129]. The result was that the working array was kept in a default data segment referred to as DGROUP. DGROUP also contains memory pointer variables used by the compiler and run-time system; the stack, which is used for passing parameters between subroutines; static variables and constants; and addresses of other data segments such as named COMMON blocks and large arrays. The result is that the small memory model is significantly limited in comparison to the later implementation (called the large memory model) because the working array size could not drive the size of DGROUP over 64 kilobytes without declaring the dummy arrays as large arrays. The only significant difficulty encountered during this conversion was that the BLOCK DATA module would not initialize correctly. Conversations with Microsoft technical support indicated that this was a known bug and that BLOCK DATA had to appear as the first object in a link module in order for correct initialization to take place.

The correctness of the small memory model was verified with the published results used to verify the VAX

implementation. In all cases, the results obtained on the microcomputer were identical with those published by Dhatt and Tuzot [Ref. 8], with the exception of residual computations. Residual computations on the microcomputer produced number of the same (small) magnitude but not the same mantissa. It is suspected that this could be, in part, related to Microsoft's adoption of the IEEE standard for real number representation and calculations. The differences in residual computations are considered to be inconsequential by this investigator. At the time the small memory model was completed, MEF contained only the first two elements; a quadratic element for anisotropic harmonic problems in one, two, or three dimensions; and an eight noded quadrilateral element for two dimensional elasticity problems.

3. Large Memory Model Implementation of MEF

To begin with, the simplest of approaches was used to convert to a large memory model; all arrays were declared large (using the compiler "metacommand" $LARGE [Ref. 11: pp. 186-187]). This approach was used because of its simplicity, and the fact that all dummy arrays of the working array (arrays which were contained within the working array) had to have the $LARGE attribute for the compiler to generate correct linkages. Initial compiler diagnostics included errors for several DO loops that indicated that the compiler believed an illegal jump into the range of a DO had been executed. Each of the affected DO loops was nested and did

not appear to violate the specifications of FORTRAN 77. Furthermore, they had not caused problems with the implementation of the Small Memory Model. These problems were alleviated by the use of the compiler metacommand $DO66 [Ref. 11: p. 183] which tells the compiler to use the FORTRAN 66 DO loop conventions (it does not appear that this should have worked, but it did).

The resulting code compiled and linked without diagnostics but did not work. The initial symptom was that it did not recognize any of the commands contained in the input stream. Diagnostic write statements revealed that the array used to store the list of commands was not initialized correctly. The array (BLOCS) was initialized by a DATA statement within the main program. The statement appeared to be correct in syntax and generated no diagnostics, yet writing the contents of the array indicated that it contained nulls. The DATA statement initialization of the array was replaced with a call to a subroutine which initialized the array using assignment statements. This solved the command recognition problems, but runtime errors which involved a variety of arithmetic operation violations were produced. Usually these errors were overflows, underflows, or attempts to use an uninitialized variable in an arithmetic operation. The particular error depended upon what fix up had been used to overcome the previous error.

During this period, conversations with Microsoft Corporation's technical support department indicated that there were several reported bugs in the compiler. One was the BLOCK DATA problem mentioned above, and another was that arrays which had the large attribute were not always initialized correctly by data statements. However, this last problem was only supposed to occur with REAL arrays. The first problem was solved by compiling the BLOCK DATA module separately, and linking it as the first module at all times. This did not alleviated the incorrect initialization of the CHARACTER*4 command array, so the MAIN program segment was compiled separately assigning the $LARGE attribute to the working array, and allowing all other arrays in the MAIN program to default to $NOTLARGE. After this the command array was initialized correctly, but run-time errors were still a problem.

The use of character arrays to pass table names was prevalent throughout MEF, and diagnostic write statements indicated that some of them were being initialized correctly and some were not. Since there were so many of them and there did not seem to be any particular characteristic which would identify which ones would initialize and which would not, the arrays containing all of these tables were declared $NOTLARGE. This resulted in the tables initializing correctly, but the run-time errors persisted.

It is appropriate to mention here that the time to compile and link after making changes which required recompiling all modules was close to two hours on the Columbia floppy disk system. The IBM PC/XT had only recently become available and reduced compile and link time to under an hour. In addition, the linked module using the mixture of metacommands was often over 400 kilobytes long. A standard floppy diskette will only hold 360 kilobytes and these excessively large linked modules would not have been possible without the hard disk. Further, it would not have been possible to proceed without them either; the run-time errors proved essential in localizing the problems and identifying them as compiler bugs rather than logic errors. The linked module using the generic $LARGE metacommand, was approximately 239 kilobytes, and while neither module would execute correctly, the increase in size of the mixed module was highly suspect since the $NOTLARGE metacommand was supposed to produce less object code.

Further conversations with Microsoft Technical Support indicated that there had been some reports that mixing the $NOTLARGE metacommand with the $LARGE metacommand could cause problems, and that the syntax for the $(NOT)LARGE command was incorrect in the reference manual. The manual [Ref. 11] specified that the $(NOT)LARGE metacommand could be used with a string of array identifiers separated by commas. However, according to technical support each occurrence of

the metacommand could only declare a single array name, and the metacommand required a colon separator between the command and the array name (i.e., $LARGE: array). If the metacommand was used without an argument (called a generic $(NOT)LARGE) then all arrays in the compiland were considered to have the particular attribute.

Since the mixing of the $NOTLARGE command with the generic $LARGE command was suspect, numerous attempts were made to identify all arrays which required the $LARGE attribute and declare them specifically while allowing all others to default to $NOTLARGE. None of these attempts worked. Conversations with Microsoft Technical Support indicated that the investigation had possibly uncovered some new problems with the compiler and requested that the problem be documented and sent to Microsoft with a diskette containing the software (they were less enthusiastic when told how extensive the software was).

As a last act of desperation, all data statements which initialized arrays were commented out. The data statement initializations were performed with assignment statements either directly or through subroutine calls, and all modules were recompiled using the generic $LARGE metacommand. The resulting linked module was approximately 240 kilobytes long and was able to run the simple test problems with no difficulty. However, as the size of problems was increased, the behavior of the program became unpredictable.

The program worked properly until the problem required more than 64 kilobytes of the working array to run. This always occurred during execution of the assembly and solution process, and the results vary depending upon which element subroutine is being used (i.e., which incorrect internal linkage is being used).

To summarize the problems mentioned above:

1. Arrays which have the $LARGE attribute are not always initialized correctly with data statements. The incorrect initialization is not predictable, nor is it confined to REAL arrays.

2. The $LARGE and $NOTLARGE metacommands cannot be used inside the same compiland.

3. BLOCK DATA must appear as the first module in a link module.

4. Nested DO loops can sometimes generate compile time errors.

In an effort to cleanup the long streams of assignment statements caused by the data statement problem, a separate compiland was created in which subroutines whose names begin with "INIT" (see Appendix E, pp. 247-258) were placed. The method used was to pass the name of the array being initialized as a calling parameter. The passed parameter was declared $LARGE, and all other arrays were allowed to default to $NOTLARGE. A $NOTLARGE "dummy" array was initialized with a data statement and a DO loop was executed which

assigned the elements of the "dummy" array to the passed array, and then executed a return. During the creation of these subroutines it was discovered that the data statements in the first subroutine of the compiland would not initialize correctly. If the initializations were character strings no diagnostic was generated, but if the initializations were REAL constants the compiler would generate "CANNOT CONVERT CONSTANT" diagnostics. A subroutine called DUMMY which had no function, and was never called by another routine was created and placed as the first subroutine in the compiland. The method is not elegant, but it worked, and it eliminated long strings of assignment statements which did nothing more than assign constants every time a subroutine was called. In the case of many subroutines these statements were only executed once, however, in the case of element subroutines, they were executed many times during a problem solution.

During the course of this investigation five elements were added to MEF, and it was only after the implementation of these elements and the initialization routines that problems large enough to cause difficulties were attempted. It was feared that the modifications to MEF had possibly induced some of the problems. In order to demonstrate whether or not the microcomputer code was portable, and to verify that compiler bugs were the problem, and not a failure in program logic, MEF was transferred by modem to the VAX 780.

The transferred code required four lines of code to be modified. Two of the lines were OPEN statements which contained file names that had illegal character strings under the VMS operating system, and two contained an illegal format element, a backslash (in the Microsoft implementation, the backslash suppresses an automatic carriage return linefeed at the end an output line). The resulting FORTRAN program compiled and linked with no further diagnostics, and its results have been verified using published test problems [Ref. 8], and [Ref. 13: pp. 170-177]. In addition, the results have been tested using the Graphics Interactive Finite Element Timesharing System (GIFTS), and CAL-NPS. A selection of the test problems has been provided in Appendix E.

D. GLOBAL STRUCTURE AND USE OF MEF

It is not the intention of this thesis to provide a comprehensive programmer's reference manual or users guide to MEF. However, an overview of the structure of MEF will be helpful to any potential user of this powerful tool. For greater detail, the reader is referred to Chapter Six of [Ref. 8].

1. Functional Blocks of MEF

MEF consists of sixteen functional blocks. Some of the blocks are required for all problem types, and some of the blocks are optional depending on the problem being solved. The functional blocks are also the names of the

block calling cards (or commands) and are listed in Table III below. An underscore indicates that the block is required for all problem types.

Functional block diagrams are provided in Appendix C, and complete descriptions of the input data cards are provided in [Ref. 8: pp. 440-447]. The main program controls the flow of all information through the functional blocks by transferring control to a subroutine called BLNNNN when the block calling card NNNN is encountered in the input file. The subroutine BLNNNN then performs preliminary functions such as logical unit identification, and reading of control parameters for the creation of various files and tables. The subroutine then calls subroutine EXNNNN. In all cases, subroutine BLNNNN provides appropriate default parameters which will be overridden by user values if specified. Subroutine EXNNNN then performs the major operations of the block by calling on the needed subroutines in the MEF library. The above protocol holds for all blocks except STOP, COMT, and IMAG. All the functions of COMT and IMAG are performed by subroutine BLNNNN, and the function of block STOP is performed by the main program.

With the exception of blocks IMAG, COMT, and STOP each block uses a named COMMON/NNNN/ to assist in the passing of needed information between subroutines. The blocks COMT and IMAG use a named common block, COMMON/TRVL/, which is used as a scratch pad for various routines. Block STOP

does not require its own common block but uses the information from COMMON/ALLOC/ to perform its function of printing the maximum length of the working array used during execution of the problem. The common block COMMON/ALLOC/, is used by subroutine ESPACE and VIDE to keep track of the amount of working space allocated at any time. Subroutine ALLOC allocates table space, and subroutine VIDE deletes unneeded tables followed by compacting the workspace.

Variable and array naming conventions and details are contained in [Ref. 8: pp. 369-376] these details are omitted here because they will only be helpful to the reader who intends to modify MEF. In that case the reader is referred to the source for the extensive detail required.

Table III. Functional Block Summary

IMAG	Copies the input data card images to the output listing. Must be the first card if used.
COMT	Places comments into the output listing.
COOR	Reads the nodal coordinates and number of degrees of freedom of each node. Provides automatic node generation.
DLPN	Provides the ability to modify the degrees of freedom at a node. Particularly useful with problems using more than one element type.
COND	Reads the boundary conditions.
PRND	Reads nodal properties if required by the problem.
PREL	Reads element properties if required for the element type being used.

ELEM	Reads the element connectivities. Also reads element group information when more than one element type is used, or when elements have different properties. Provides automatic element generation.
SOLC	Input of concentrated loads.
SOLR	Input of distributed loads.
LINM	In-core assembly and solution of a linear system of equations.
LIND	Out-of-core assembly and solution of a linear system of equations.
NLIN	Provides a limited nonlinear solution capability using the Newton-Raphson method.
TEMP	Provides the solution of a linear or nonlinear time dependent problem using an Euler method.
VALP	Computes eigenvalues and eigenvectors using the sub-space iteration method.
STOP	Terminates execution of the problem.

The following information concerning array names is provided because the array names may appear in the output listings with no explanation when verbose printouts are requested. Block COOR creates the table of nodal coordinates in the array VCORG, and the cumulative degrees of freedom in the array KDLNC. Block COND stores the equation identification number for each degree of freedom in the array KNEQ, and the specified degrees of freedom at a boundary in the array VDIMP. Block ELEM creates the array KLD which contains the location of the beginning of each column in a skyline matrix, and writes a disk file containing all information pertinent to the description of an element. The disk file will be used

in the assembly process to create element and global stiffness matrices. Block PREL creates the array VPREG in which it stores the properties of the various groups of elements. The global stiffness matrix, VKG, is created in block LIND or LINM. The entire stiffness matrix consists of three submatrices VKGS, VKGD, and VKGI which contain the upper triangle, the diagonal, and the lower triangle of the matrix, respectively. VKGI is only present for nonsymmetric matrices. The Matrix VFG contains the global loads, VDLG is the solution vector, and VRES is the residuals and reactions vector.

As mentioned above, MEF provides various levels of output. The quantity of output desired from a given block is controlled by a parameter on the block calling card (described in detail [Ref. 8: pp. 440-447] which ranges from 0 (the assumed value) to 4. The default value provides all the information needed to verify the input stream and obtain the desired answers while the values 1 thru 4 provide various levels of verbosity. Each level provides all that the last one did plus additional information, and in some blocks levels above 2 have no additional meaning. The reader is cautioned against the indiscriminate use of verbose listings. With larger problems it is easy to create megabytes of listing which can even overrun the capacity of a hard disk. The best procedure is to decide where verbose output is needed and to use it only in the required

blocks. Using verbose printing on the solution blocks creates reams of output and is seldom of any value to the user. The solution processes have all undergone extensive verification, and problems which arise are generally traceable to the input stream. The only exception is when the problem requires more than 64 kilobytes of the working array in which case the program will most often terminate with a run-time arithmetic operation error. In this case, further processing will have to be done with the VAX 780 version.

2. Elements Supported by MEF

The current version of MEF consists of approximately 7000 lines of code (including comments). Five additional elements have been added to make a total of seven elements. A summary of the elements is listed in Table IV. All elements have been written in French and with the exception of elements 1 and 2 produce most of their output in French. The user will have little if any trouble understanding the results and should consider the experience culturally enriching. Most of the words which appear in the listings are cognates or recognizable from the context in which they appear. Time did not permit the translation of the format statements.

The block PREL will require the properties of the elements to be entered as a data card image. The properties are specific to the element routine alone, and must be

entered in the order expected by the element routine. Table IV summarizes the properties required by each element in the correct order. For those elements which require a material density property, the property is used in the creation of a mass matrix for the solution of eigenvalue problems. This property may be omitted if the block VALP will not be used. Element 5 is not implemented in this version, and has been left out of Table IV.

Table IV. Element Summary

ELEMENT NUMBER	DESCRIPTION	REQUIRED PROPERTIES
1	Eight noded quadrilateral for anisotropic harmonic problems in 1, 2, or 3 dimensions	1) coefficient DX 2) " DY 3) " DZ 4) specific heat capacity
2	Eight noded quadrilateral for 2 dimensional elasticity problems	1) Young's modulus 2) Poisson's ratio 3) 0 = plane stress 4) specific mass
3	Six noded triangular element for 2 dimensional elasticity problems	1) Young's modulus 2) Poisson's ratio 3) 0 = plane stress 1 = plane strain
4	Three noded triangular element for 2 dimensional regions of unit thickness	1) Young's modulus 2) Poisson's ratio 3) 0 = plane stress 1 = plane strain 4) X body force component 5) Y body force component 6) specific mass

ELEMENT NUMBER	DESCRIPTION	REQUIRED PROPERTIES
6	Three noded triangular plate bending element for isotropic or orthotropic materials Notes: If the material is isotropic properties are 5) Young's modulus 6) Poisson's ratio 7), 8) = 0 If orthotropic 5) D(1,1) 6) D(1,2) 7) D(2,2) 8) D(3,3) where the D(i,j) are the bending stiffness elasticity constants	1) index for integration by Gauss-Radau (1-5) 1 = least accurate 5 = most accurate 2) thickness 3) 1 = isotropic 2 = orthotropic 4) location to calculate stresses 1 = centroid 2 = corner nodes 3 = midnodes 5) - 8) according to notes 9) specific density
7	Twenty noded brick for three dimensional elasticity problems	1) Young's modulus 2) Poisson's ratio
8	Truss element for 2 or 3 dimensional problems	1) cross-section 2) Young's modulus 3) density

3. Running MEF

To execute MEF as installed on an IBM PC-XT or compatible, simply boot the operating system, log to the directory in which MEF.EXE is located, and type MEF followed by a carriage return. MEF will respond by asking for the name of the command input file. The response may be any legal MS-DO5 file name, including a disk drive identifier (for example, a:INPUT.DAT). If the response is the MS-DOS identifier CON then MEF will expect to receive all commands and inputs from the console keyboard. After entering the

input file, MEF will request the name of the output file. Once again, this may be any legal file name including PRN. The response PRN will result in the output being directed to the printer. MS-DOS pathnames are not supported by MEF in the naming of input and output files. After the entry of the output file, MEF will begin to process the command input file; as MEF processes the input commands it will update the console with information concerning which functional block it is processing.

When MEF is used on a system which does not have a hard disk it is best to keep the input and output data files on a separate diskette in the default drive, and execute MEF from the default drive using a drive designation. For example, if the default drive was a: the MEF program could be started by typing b:MEF with the MEF diskette in drive b: and a scratch diskette in a:. The reason for this is that MEF creates several scratch files on the default drive during execution, and there is not much room left on a floppy diskette which contains MEF. MEF will create two scratch files for an in-core solution, and three scratch files for an out-of-core solution; the names of these files will begin with $$ so that it is not likely they will coincide with existing file names.

Once the initial responses to MEF have been made, the present version of the code expects all input to come from a command file, or the console. Attempting to input

directly from the keyboard can be a very frustrating experience, and if a mistake is made there is no recourse but to begin again. For this reason, it is recommended that a command input file be created with a suitable text editor. FORTRAN 77 rules apply to the format of the cards. That is, entries separated by commas will override the specified format, however, there must be no blanks imbedded in the line if the format is to be overriden. As can be seen from the examples provided in Appendix D either method will work, and the user may find it convenient to enter some cards according to the format, while other cards may be easier to override.

It is advisable to send the output file to disk rather than to the printer. The disk file can be viewed and even edited with a text editor prior to printing. Sending the output to the printer will simply cause the process to be output bound. MEF also uses a 132 character output line, and it is advisable to shift the printer to a 17 character per inch mode if it does not have a wide carriage.

The amount of space required to run the problem must be of concern to the user until the bugs have been eliminated from the compiler. The required number of bytes may be estimated using the following formula:

space required = (bandwidth)(number of nodes)(8)(2.0)

The factor of 2.0 is an empirically determined factor used to account for the storage of all tables in the working array. Because of the compiler bugs which have not yet been circumvented or corrected, if the space required approaches 64 kilobytes then the compiler/IBM PC capabilities have been exceeded, and MEF will probably fail with a run-time error.

III. RESULTS AND CONCLUSIONS

A. SUMMARY

Chapter I recounted a portion of the history and development of the microcomputer and attempted to list some of the reasons that microcomputer based, engineering software has been slow to develop. The main reason is that it was too difficult to create engineering software on the rather limited resources provided by the eight bit microprocessor, and limited software tools which existed at the time. In addition, the engineering software which was created was slow and unwidely to use which hampered its propagation and development. However, the advent of the sixteen bit microprocessor has provided a hardware product whose capabilities are more than adequate for engineering applications. This opinion is supported strongly by the fact that the majority of main-frame minicomputer systems today are based on sixteen bit processor architecture. The major difference between the minicomputer and the microcomputer is operating system maturity, and processor speed. The speed advantage is partially offset by the fact that a microcomputer is seldom used to support timesharing applications, and can often produce results almost as quickly as the minicomputer burdened with the management of timesharing (one must count the time spent waiting, not just the CPU seconds).

Regardless, the sixteen bit microcomputer has been around since 1981, yet there is little engineering software available today. The reason is that reliable software tools (operating systems, compilers, etc.) lag the introduction of hardware by a considerable amount of time. The premise of this thesis was that the necessary maturity of operating system and compiler had been achieved, and a combination of hardware, operating system, and compiler was chosen to test the premise in the specific application of finite elements.

As stated in Chapter I.B, the purpose of this investigation was to implement a general, finite element program on a sixteen bit microcomputer, and determine whether the result was practically useful. The actual programming and conversion of software began in March 1984, and continued thru August 1984. During that period two distinct version of MEF were installed on the IBM PC-XT and the COLUMBIA MPC microcomputers. The first version was a small memory model which performed all the functions of the mainframe version but was quite limited in the size of problems it could handle. This was to be expected, and was merely a point in the stepwise implementation of the objective.

The small memory model was then converted to the current version of MEF which is referred to as the large memory model. The large memory model is significantly more capable than the small memory model in terms of the problem size that can be handled. However, as detailed in Chapter II the large

memory model is not able to take advantage of the full memory available because of the existing bugs in the Microsoft FORTRAN 77 Compiler version 3.2.

B. CONCLUSIONS

Although it was not the intention of this investigation to evaluate the hardware or operating systems of the two machines, it is impossible to write a conclusion without mentioning them. Throughout this investigation, both systems (the IBM PC-XT, and the COLUMBIA MPC) have functioned faultlessly. This observation includes the operating systems and the hardware. Both computers have been supported by a variety of peripherals manufactured by different companies, and neither system has operated in a controlled environment. The machines are turned off and on at will, and have received only the most cursory preventive maintenance. Yet, both systems have maintained one hundred percent availability, on demand, with no time spent at reduced capability. The previous experience of this investigator has been with mainframe computing systems, and the reliability of these microcomputer systems was totally unexpected.

At the beginning of this investigation, it was not clear that a program the size and complexity of MEF could be converted to operate on a microcomputer. However, a background in computer science and operating systems led this investigator to believe that it might be possible. The results have been

gratifying on one hand, and frustrating on the other. The frustration results from the limitations imposed, not by the program, and not by the hardware, but by the immaturity of the compiler. With the example of the matrix solver shown in Appendix B it is clear that the machine and compiler combination have the capability to solve large problems. It is unfortunate that compiler bugs prevent the full realization of that capability with a more complex application. However, conversations with Microsoft Corporation indicate that a new release of the compiler may be available as early as January 1984, and even at this time, advertisements for competitive compilers are beginning to appear in periodicals.

A more objective statement of the results is that the largest problem which could be run on the microcomputer took less than five minutes from start to stop, and the results are comparable to the results obtained from other sources. It is clear that the execution speed and capability of the software is acceptable. Therefore, the utility of MEF is assured, subject to the temporary restriction of problem size. At this time, MEF is an excellent classroom tool, and is capable of solving most problems given as academic exercises in solids and conduction heat transfer. It is also capable of handling many problems which are not assigned as academic exercises. In addition, because of its modular structure, MEF also provides an excellent teaching tool for

the finite element classroom. As soon as the problem size restrictions are overcome, MEF will have far greater application on the microcomputer.

It is important to understand the significance of what the ability to create and execute software of this complexity and capability (on a microcomputer) can mean to the field of engineering in general. If the compiler had been "clean," the problems encountered in converting MEF would have been minimal. The cost of a microcomputer system is well within the range of most small engineering firms, and the increase in problem solving capability is even more dramatic than the step from the sliderule to the programmable, pocket calculator. There is a wide variety of software available today including finite elements, optimization, heat transfer, fluid dynamics, electronic circuit design, control systems, etc. The cost of computer time has made much of this software unavailable to smaller concerns. However, the near future will undoubtedly see the conversion of much of this software to microcomputer systems. The possibilities are encouraging.

C. RECOMMENDATIONS

The following recommendations are made for future development of MEF:

1. MEF, as implemented, is primarily a batch stream processor. By that it is meant that the input is noninteractive and formatted. The facility of MEF would be enhanced by

the addition of interaction, or an interactive preprocessor to produce the "steering file" (command input file).

2. The capabilities of graphics to summarize the results from any finite element application cannot be over stated. In addition, the graphic representation of the structure and finite element mesh is important for the detection of errors in the problem definition. Therefore, the addition of graphics to MEF would significantly improve its capability.

3. The possibilities regarding the addition of elements are almost without bound. However, the addition of a cubic solid element (a 32 node brick) would provide significant additional capabilities. The addition of such an element would provide an exact solution for beams (using only one element for node loadings), and an excellent model for plates and shells. The addition of such an element would allow the elimination of a number of the existing elements, at the cost of more memory; the trade off would have to be evaluated.

LIST OF REFERENCES

1. Rao, Guthikonda V., Microprocessors and Microcomputer Systems, 2nd ed., pp. iii-8, Van Nostrand Reinhold Company Inc., 1982.

2. Mulholland, David Joseph, An Investigation of the Feasibility of Implementing Substantial Finite Element Codes on Popular Microcomputers, Master's Thesis, Naval Postgraduate School, 1982.

3. Mallory, Ray R., A Finite Element Program Suitable for the Hewlett-Packard System 45 Desktop Computer, Master's Thesis, Naval Postgraduate School, 1980.

4. Tetewsky, Avram, "Benchmarking FORTRAN Compilers," BYTE, v. 9, no. 2, pp. 218-222, February 1984.

5. Sarnak, Neil and Jaffe, Eric, "8087 Performance Considerations," PC Tech Journal, v. 1, no. 2, pp. 30-46, September-October 1983.

6. Wilson, Edward L., Structural Analysis on Microcomputers, paper presented at Conference on Finite Elements using Microcomputers, Italy, April 1984.

7. Rector, Russell and Alexy, George, The 8086 Book, pp. 4-1 thru 4-70, Osborne/McGraw-Hill, 1980.

8. Dhatt, Gouri and Touzot, Gilbert, The Finite Element Method Displayed, John Wiley & Sons, 1984.

9. Microsoft Corporation, Microsoft FORTRAN Reference Manual, 1983.

10. Microsoft Corporation, Microsoft FORTRAN Users Guide, 1983.

11. Microsoft Corporation, Microsoft FORTRAN Reference Manual, 1984.

12. Microsoft Corporation, Microsoft FORTRAN Users Guide, 1984.

13. Felippa, Carlos A., Refined Finite Element Analysis of Linear and Nonlinear Two-Dimensional Structures, University of California at Berkeley, October 1966.

APPENDIX A

CPU CLOCK SPEED TEST

The following Intel 8088 Assembly language program, assembled with the MICROSOFT assembler (MASM), was used to determine the apparent speed of the processor, or the loss of processor speed due to the processing involved in handling the interrupt driven operating system.

When executed, the program loops for the number of times specified. The actual number of times through the loop can be determined by multiplying the contents of the BX register by 65,536. In this application, the result is 10,485,760. The program accesses the system clock before and after completion of the loop, and computes the elapsed time to the nearest second. The elapsed time is then displayed on the screen. The test was done numerous times on both systems, and never computed an elapsed time less than 52 seconds nor one greater than 53 seconds.

Clock cycle calculations were computed by counting the total number of machine cycles executed between the labels WAIT and ENDWAIT. The eight cycles used for initialization in the two steps before the label WAIT have been included only for the sake of preciseness so that all machine cycles between clock accesses were accounted for. The number of

times through the loop is large enough to insure that any error induced by not counting the clock accesses is insignificant.

INSTRUCTION	NO. OF TIMES EXECUTED	CLOCK CYCLES PER EXECUTION	NO. OF CYCLES IN LOOP
MOV BX,0A0h	1	4	4
MOV CX,00h	1	4	4
DEC BX	160	2	320
JZ ENDWAIT	1 XFER	16	16
	159 FAILS	4	636
DEC CX	1048576	2	20971520
JNZ LOOPS	1048560 XFERS	16	167769600
	160 FAILS	4	640
JMP WAIT	160	15	2400
	Total Clock Cycles In Loop		188745140

$$\text{APParent Clock SPeed} = \frac{188745140}{53} = 3.56 \text{ MHz}$$

```
CSEG    SEGMENT    PARA 'CODE'
        ASSUME     CS:CSEG, DS:CSEG, SS:STACKSG, ES:NOTHING
;
;
        ORG     0100H
        PUSH    DS              ;Save DS for return to DOS, and
        SUB     AX, AX          ;put a zero on the stack.
        PUSH    AX
        MOV     AX, CSEG        ;Set the DS register.
        MOV     DS, AX
;
;
        LEA     DX, STARTMSG
        CALL    OUTMSG          ;Output start message
                                ;to screen.
;
;
        CALL    BEEP            ;Beep terminal bell.
;
;
        CALL    GETTIME         ;Reads clock chip and stores
        MOV     STARTTM, DX     ;minutes and seconds in memory
                                ;location, STARTTM.
;
;
        MOV      BX, 0A0h       ;Initialize counters for delay loop.
        MOV      CX, 00h        ;For real run BX=0A0h
;
;
WAIT:   DEC      BX             ;Run around in circles about
        JZ       ENDWAIT        ;10 million times.
LOOPS:  DEC      CX
        JNZ      LOOPS
        JMP      WAIT
;
;
ENDWAIT:
        CALL     GETTIME        ;Read the clock chip and
        MOV      STOPTM, DX     ;store in memory location STOPTM.
;
;
        CALL     BEEP           ;Beep terminal bell.
;
;
        LEA      DX, ENDMSG
        CALL     OUTMSG         ;Send all done message
                                ;to the terminal.
;
;
;ELAPSED TIME                    Compute elapsed time.
```

```
                                ;Elapsed time is assumed less than
                                ;one minute.
        XOR     AX, AX          ;Clear AX.
        MOV     BX, STOPTM
        MOV     CX, STARTTM
        MOV     AL, BL          ;Stop time in seconds in AL
        CMP     BH, CH          ;If minute has incremented during
        JE      LBLA            ;wait loop must add 60 seconds to
        ADC     AL, 60h         ;stop time to compute correct delta t.
        DAA                     ;All of this works because the clock
LBLA:   SUB     AL, CL          ;provides BCD quantities.
        DAS
        MOV     DX, AX
;
;
        CALL    ASCCONV         ;Convert elapsed time to ASCII,
        LEA     DX, ELTIMEMSG;and output elapsed time, in seconds,
        CALL    OUTMSG          ;to the screen.
;
;
        MOV     AH, 4Ch         ;Return to DOS.
        INT     21h
;
;
BEEP    PROC    NEAR
        MOV     AH, 02h         ;SUBROUTINE to beep the
        MOV     DL, 07h         ;terminal bell.
        INT     21h
        RET
BEEP    ENDP
;
;
GETTIME PROC    NEAR
        MOV     DX, 02C2h       ;SUBROUTINE to reads system clock.
        IN      AX, DX
        MOV     DX, AX          ;The hours are placed in CX, and the
        RET                     ;low order count (approx 18.2 counts
GETTIME         ENDP            ;per second) in DX.
;
;
OUTMSG  PROC    NEAR
        MOV     AH, 09h         ;SUBROUTINE to output string pointed
        INT     21h             ;to by DX.
        RET
OUTMSG  ENDP
;
;
ASCCONV         PROC    NEAR ;Convert elapsed time to ASCII.
        LEA     SI, ASCVAL+3 ;SI points to least significant
                                ;digit's storage location.
        MOV     CX, 04          ;Initialize loop counter.
```

```
LBLC:   PUSH    CX              ;Save the loop count.
        MOV     CX, 04          ;Shift count in CX.
        AND     AX, 000Fh       ;Strip right most nybble.
        OR      AX, 30h         ;Convert digit to ASCII character.
        MOV     [SI], AL        ;Store the character.
        DEC     SI
        MOV     AX, DX
        SHR     AX, CL          ;Move the next digit into the least
        MOV     DX, AX
        POP     CX
        LOOP    LBLC            ;significant nybble.
        RET
ASCCONV ENDP
;
;
STARTMSG  DB    'Begin wait loop',0Dh,0Ah,0Ah,0Ah,'$'
ENDMSG    DB    'End wait loop',0Dh,0Ah,'$'
STARTTM   DW    ?
STOPTM    DW    ?
ELTIMEMSG DB    'Elapsed time in seconds:  '
ASCVAL    DB    '    $'
;
;
CSEG      ENDS
;
;
STACKSG   SEGMENT  PARA STACK 'STACK'
          DW    80 DUP(?)
STACKSG   ENDS
          END
```

APPENDIX B

MATRIX SOLUTION TEST PROGRAM

The following FORTRAN program was used to test the capability of the machine to solve a system of equations, [A](X) = (B), whose coefficient matrix, [A], required more than 65,536 bytes of storage. The main program requests a job name, and the number of equations to be solved. It then fills the [A] matrix symmetrically, in banded fashion, with the number of equations (NEQ) on the diagonal, and each subdiagonal decreased by one more than the previous subdiagonal; the right hand side of the system, (B), is always a vector of 100.0's. For example, if the number of equations were 5 the program would solve the following system:

$$\begin{matrix} 5 & 4 & 3 & 2 & 1 \\ 4 & 5 & 4 & 3 & 2 \\ 3 & 4 & 5 & 4 & 3 \\ 2 & 3 & 4 & 5 & 4 \\ 1 & 2 & 3 & 4 & 5 \end{matrix} \quad \begin{matrix} X1 \\ X2 \\ X3 \\ X4 \\ X5 \end{matrix} \quad = \quad \begin{matrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{matrix}$$

The program stores the solution and the solution time on the disk in a file which is identified as jobname.DAT. In addition, the results and solution time are displayed on the console. The system is positive definite which guarantees

that no processor error condition will occur in the solution of the system. The number of steps to achieve solution is fixed for a given matrix size, and the amount of time to achieve the solution is not affected by the accuracy of the answers. In short, the only thing which is of interest here is being able to run a series of benchmarks which are guaranteed to proceed to completion.

The elapsed time is determined by successive calls to an assembly language routine, TICKER, which must be assembled separately and linked to the FORTRAN subroutines. The routine was added because the Microsoft FORTRAN compiler has no function which allows access to the system clock. For large systems, which require more than a few seconds to solve, the program could easily be set up to signal the user to start and stop timing with a stop watch. However, for smaller systems, such as the 25 and 32 DOF tests, the elapsed time is too small to determine with a stopwatch. Particularly when the results are to be used to predict solution times for larger systems.

The results of the four test runs are as follows:

```
                       FOR TEST-1.DAT
                       The Solution Is
                       ===============
 3.8461538461538E+00   6.9269455534349E-15  -5.8183935461887E-15
-1.5539431443739E-15   6.9501601786561E-15  -7.3976511644819E-15
 6.7087227108802E-15  -1.2678408065005E-15  -4.8174716530773E-15
 1.3505456915410E-15   4.4550084107864E-15  -5.7056834328442E-15
-4.0948604156610E-17   7.4119342420388E-15  -7.3233038075074E-15
 1.6435513664323E-15   6.7101591338120E-15  -2.0836013365003E-14
 2.7453422022684E-14  -1.9176192891666E-14  -2.6964397150234E-15
 2.1610766337363E-14  -9.5897682156343E-16  -2.1338961976026E-14
 3.8461538461539E+00

   TIME =       .05767 MINUTES


   NEQ  =           25
```

```
                       FOR TEST-2.DAT
                       The Solution Is
                       ===============
 3.0303030303030E+00   2.8196140307940E-17   3.1618483033404E-16
-5.3765407995495E-16   8.6583481992882E-16  -2.8342882419572E-15
 2.0606496035387E-15   7.9685816303165E-16   2.1751157329228E-15
-6.7609270454854E-15   2.0326481785032E-15   5.7594701413748E-15
-5.7766604500654E-15   3.3827107781548E-16  -8.0238398139056E-16
 2.3465232733079E-15   1.2883032314559E-14  -2.2432853447170E-14
 1.2880062543391E-14  -3.9098450557409E-15   2.1589175759618E-15
-8.0694495013671E-15   2.1399342284950E-14  -2.4628800015674E-14
 2.9480758112488E-15   2.2982194850899E-14  -2.0530161315457E-14
-4.9194647979880E-15   2.4517690154337E-14  -2.4163037642173E-14
 9.7680238081105E-15   3.0303030303030E+00

   TIME =       .11800 MINUTES


   NEQ  =           32
```

FOR TEST-3.DAT

The Solution Is

==============

9.9009900990099E-01	9.5453377195709E-17	-6.8540742491008E-16
6.6510189646372E-16	3.1602280670458E-17	-2.1833385348956E-16
6.0231650128169E-15	-7.5460757704011E-15	1.8959559151423E-15
-2.0094637123553E-15	2.4533840328452E-15	-1.4785018355949E-16
-2.8687989483966E-16	-1.4755330610054E-15	5.8507425541527E-16
-1.1568290666613E-15	1.5067832271187E-15	3.0463592713988E-16
8.7207840465378E-15	-1.2381694833975E-14	5.8759339624861E-15
-1.6390411200061E-15	-5.2001533979711E-15	1.1056030982832E-14
-1.2004944675362E-14	6.8472608535533E-15	-4.1598391672334E-15
8.1980386773411E-15	-1.0210186519736E-14	2.2705640872189E-15
-4.1109406779441E-16	9.5382686124413E-15	-1.1086024973244E-14
1.2843962732201E-14	-1.1193652655344E-14	-8.8145373786180E-16
4.1975035855025E-15	6.4546905439846E-15	-1.6955941176263E-14
2.0954050800393E-14	-9.0052087558773E-15	-1.2414564921144E-14
3.8276714669663E-15	1.2806988307790E-14	5.8470481061057E-15
-9.5233129178282E-15	-7.7232684139203E-16	-1.4190569505143E-14
3.4023789901836E-14	-3.0417698345392E-14	1.3110805120962E-15
5.4919387864296E-15	-2.6434621782263E-15	1.9542460928892E-14
-1.9756178267955E-14	3.9992584196400E-15	2.1280027556825E-15
-2.5651465217669E-16	-1.3820477797203E-15	1.3528536299486E-15
-9.8171447719577E-15	2.8267623241653E-14	-2.6361915038062E-14
1.7386323809334E-14	2.1753926657387E-15	-5.4039187890453E-14
7.4810453570045E-14	-3.7721279929513E-14	5.8007280323821E-16
-5.2844278730221E-15	9.2214384665882E-15	2.8665906218786E-14
-4.9256092013141E-14	2.6317476194804E-14	-1.7483217375341E-14
5.3017426386583E-14	-5.0804380408870E-14	-4.8311343633811E-17
1.3768713958846E-14	-2.5653900778252E-14	5.1317810949042E-14
-5.5388663718596E-14	4.7481349640073E-14	1.4264847983392E-15
-4.7976771901982E-14	-5.4655800567770E-15	4.5790962529292E-14
1.0906959678161E-14	-4.9474940339240E-14	2.5633078930214E-14
2.3084307311326E-14	-7.4740551015651E-14	8.1039111510772E-14
5.5017140084367E-15	-7.2903626883754E-14	1.4338852851700E-14
3.6541830133710E-14	-8.2333428774833E-15	1.7611461206863E-15
9.9009900990098E-01		

TIME = 3.47767 MINUTES

NEQ = 100

FOR TEST-4.DAT

The Solution Is

==============

4.9751243781094E-01	8.1040715769944E-16	-6.3022704105503E-16
-4.5243592798550E-17	4.8230247668559E-16	-4.8094686856488E-16
1.9287997925250E-16	-4.6407798043720E-16	1.1736265177040E-15
-2.9397240715880E-16	-1.5759067617456E-15	1.9859893106254E-15
3.5684193113147E-15	-2.7160690412767E-15	-3.2300531460591E-15
2.2367031680076E-15	-7.8610974959264E-16	-1.0140658982675E-15
2.3423274737468E-15	-1.2265976148990E-15	-9.4310203321462E-16
-2.3369133923859E-15	4.3545595605503E-15	-4.2681401557947E-16
1.4815347311197E-15	-2.8874830767152E-15	-4.3348156712426E-16
1.6098180082962E-15	-6.7681167605003E-16	-3.3024941370422E-15
4.1142527518302E-15	2.5555952712168E-16	-4.2330839814229E-16
-1.3439971017726E-16	-1.0247340536123E-15	5.9525417200927E-16
3.6405878180938E-15	-2.6210766726808E-15	-4.1367853692529E-17
-7.4204493566424E-16	2.0984858032874E-15	1.4503544604859E-15
-5.5831197567108E-15	-1.0008036406686E-16	4.3987323183762E-15
-1.1138016247761E-14	9.9029445306656E-15	8.0220946482954E-15
-1.2731667911900E-14	1.9041184820711E-15	-1.6908056985759E-15
1.1129991569071E-14	-9.4719234613579E-15	-4.0436863526805E-15
7.8544510136269E-15	1.9739929108076E-15	-2.7928685511919E-16
-1.8908238702695E-14	2.5020628162063E-14	-7.9476996716795E-15
-2.6984423267922E-15	8.4713928861782E-16	6.0303129965946E-15
-1.2559422095201E-14	1.5484553485252E-14	-1.5803636384612E-14
8.9192780343309E-15	-2.9080259039352E-15	-1.1540290295994E-14
1.6831028417556E-14	-3.4373309123280E-15	9.8243175602359E-15
-1.4528803365360E-14	1.2599790630863E-14	-1.1639838794459E-14
-4.0419987736125E-15	-9.7892957149773E-16	8.8905487799502E-15
6.7143807187873E-16	-3.2914241639296E-15	2.0976408578060E-14
-4.3836842898259E-14	4.0763982358910E-14	-1.4494634544272E-14
-1.7159643252139E-14	1.4877203649456E-14	4.5129255067718E-15
9.5041349490937E-15	-2.3408226882068E-14	3.4801730930349E-14
-5.1536403462991E-14	3.1699186457987E-14	-2.4605638291722E-15
-1.7625828454096E-14	3.5068528304064E-14	-2.6191602352315E-14
8.2644581147134E-15	-5.4712634542109E-16	7.7051044258022E-16
-1.3568562182020E-14	2.3573520522775E-15	1.6373963874779E-14
-2.2276578855938E-14	3.4438832352252E-14	2.0440983681865E-16
-3.1446321094405E-14	-1.8711978854088E-15	2.1665225141320E-14
-7.6312355214359E-15	-1.5691429652028E-14	1.8378067173779E-14
1.1693870743178E-15	1.1510062775483E-14	-8.2096513820525E-15
-2.5021610665137E-14	2.4449298808106E-14	-4.8052611442444E-14
8.8949331557590E-14	-6.0429388230552E-14	1.3868174870083E-14
4.8018624008535E-15	-3.6974325184603E-14	4.0256625174786E-14
2.3609456951757E-14	-4.8371859447327E-14	7.3210921226879E-15
-5.4417749168070E-15	1.9555839971006E-15	5.7753031997314E-14
-9.8757298592470E-14	5.7982607852308E-14	3.1834614231439E-15
4.2798877724721E-15	-2.4817562337530E-14	-2.2018042934005E-14
7.3705548019714E-14	-1.6894473575267E-14	-6.3007982169033E-14
4.1742722696338E-14	1.5015125441023E-14	-7.8160903981022E-15

-2.2835011964091E-14	1.2689163938884E-14	4.6640221501102E-14
-1.0022774539354E-13	1.1574453269464E-13	-9.6433876688429E-14
4.3343846624364E-14	-3.6739450892136E-14	1.3493315031049E-14
7.0939813268461E-15	2.0012673356105E-14	-1.2909697895391E-14
2.8251919858608E-14	-6.4663404204362E-15	-1.4025495529011E-14
-3.2055878018689E-14	1.1309775543359E-14	1.9056404665781E-14
-1.5764743391028E-14	2.3657361537963E-14	-1.2567799398804E-14
-4.2716861775601E-15	-1.8147460594876E-15	-4.0141086848358E-14
1.1405791906399E-13	-5.5741573151303E-14	-4.2469852155577E-14
-4.2453076345260E-15	1.0362532985526E-13	-8.8225045474678E-14
2.9451463111826E-14	-2.8870848145435E-14	3.3101567329672E-14
-6.8346372628696E-14	7.3991561334720E-14	4.4397262181506E-15
-5.7453411423013E-14	1.8413394245291E-14	2.4737510482256E-14
4.1497854538118E-14	-5.9210679251088E-14	-2.6246245089131E-14
6.9963930172649E-14	-5.3643566057255E-14	-6.0964496532426E-15
2.6881322987547E-14	4.7034090288469E-14	-6.1331586904634E-14
-1.6406994082191E-14	1.9008330582495E-15	6.5605392492665E-14
-7.9561942476198E-14	1.3378275230445E-13	-1.3124826477839E-13
-9.1658330019724E-15	8.5868721046333E-14	-9.8177146190048E-14
8.4360246180512E-14	4.9751243781092E-01	

TIME = 27.56250 MINUTES

NEQ = 200

Program Listing

```
$LARGE
$NOFLOATCALLS
      PROGRAM SOLDP
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER*4 ITIME1, ITIME2, CENTI
      DIMENSION A(40000), B(200), JOBNAME(2)
      CHARACTER*12 IFN
      REAL*4 TIME
      OPEN(5,FILE='CON')
      OPEN(6,FILE='CON')
      WRITE(6,*) ' ******* ENTER A JOBNAME (8 CHARACTERS MAX) **'
      READ(5,10) JOBNAME
 10   FORMAT(2A4)
      WRITE(6,*) ' ******* ENTER THE NUMBER OF EQUATIONS *******'
      WRITE(6,*) '      (MUST BE 200 OR LESS IN THIS VERSION)     '
      READ(5,*) NEQ
      RHS=100.0D0
      IEXT='DAT'
      CALL FNAME(JOBNAME,IEXT,IFN)
      OPEN(2,FILE=IFN,STATUS='NEW',FORM='FORMATTED')
      NEQM1=NEQ-1
      DO 100 I=1,NEQM1
      B(I)=RHS
      II=(I-1)*NEQ+I
      A(II)=NEQ
      IP1=I+1
      DO 100 J=IP1,NEQ
      IJ=(J-1)*NEQ+I
      JI=(I-1)*NEQ+J
      A(IJ)=NEQ-J+I
      A(JI)=A(IJ)
 100  CONTINUE
      B(NEQ)=RHS
      NEQNEQ=NEQ*NEQ
      A(NEQNEQ)=NEQ
      CALL TICKER(ITIME1)
      CALL ELU(A,NEQ)
      CALL SLVB(A,B,NEQ)
      CALL TICKER(ITIME2)
      CENTI = ITIME2 - ITIME1
      TIME = FLOAT(CENTI) / 6000.
      WRITE(2,998) IFN
      WRITE(6,998) IFN
 998  FORMAT(T28,'FOR ', A12,/,
     *       T28,'The Solution Is',/,T28,15(1H=))
      DO 200 I=1,NEQ,3
```

```
      JJ=I
      JJP=JJ+2
      IF(JJP.GT.NEQ) JJP = NEQ
      WRITE(2,'(1P3E22.13)') (B(J),J=JJ,JJP)
      WRITE(6,'(1P3E22.13)') (B(K),K=JJ,JJP)
  200 CONTINUE
      WRITE(6,999) TIME
      WRITE(2,999) TIME
  999 FORMAT(/,5X,' TIME = ',F12.5,' MINUTES',//)
      WRITE(2,*) '     NEQ  =',NEQ
      WRITE(6,*) '     NEQ  =',NEQ
      CLOSE(2)
      STOP
      END


      SUBROUTINE ELU(A,N)
C
C**********************************************************************
C                                                                     *
C    THIS SUBROUTINE DECOMPOSES MATRIX A INTO A LOWER UNIT            *
C    TRIANGULAR AND AN UPPER TRIANGULAR MATRIX. THE ORIGINAL MATRIX   *
C    A IS REPLACED BY THE TWO TRIANGULAR MATRICES. THE DIAGONAL OF    *
C    THE LOWER MATRIX IS NOT NEEDED SINCE IT IS A UNIT TRIANGULAR     *
C    MATRIX. THIS IS A MODIFICATION OF A SUBROUTINE WRITTEN IN        *
C    1965 .                                                           *
C                                                                     *
C**********************************************************************
C                                                                     *
C               COPYRIGHT (C) BY GILLES CANTIN                        *
C             MONTEREY, CALIFORNIA,  24 JULY 1984.                    *
C                                                                     *
C**********************************************************************
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(1)
      NM1=N-1
      DO 100 K=1,NM1
      KP1=K+1
      KK=(K-1)*N+K
      AKK=A(KK)
      DO 100 I=KP1,N
      IK=(K-1)*N+I
      G=-A(IK)/AKK
      A(IK)=G
      DO 100 J=KP1,N
      IJ=(J-1)*N+I
      KJ=(J-1)*N+K
  100 A(IJ)=A(IJ)+G*A(KJ)
      RETURN
      END
```

```
      SUBROUTINE SLVB(A,B,N)
C
C**********************************************************************
C                                                                     *
C   THIS SUBROUTINE DOES A BACKWARD SUBSTITUTION FOLLOWED BY A        *
C   FORWARD SUSBSTITUTION OF B INTO A, WHERE A HAS ALREADY BEEN       *
C   DECOMPOSED BY A CALL TO ELU. THE VECTOR B IS DESTROYED AND        *
C   REPLACED BY THE ANSWERS TO THE SYSTEM OF LINEAR EQUATIONS.        *
C                                                                     *
C**********************************************************************
C                                                                     *
C             COPYRIGHT (C) BY GILLES CANTIN                          *
C            MONTEREY, CALIFORNIA,  24 JULY 1984.                     *
C                                                                     *
C**********************************************************************
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(1),B(1)
      NM1=N-1
      NP1=N+1
      DO 100 K=1,NM1
      KP1=K+1
      BK=B(K)
      DO 100 I=KP1,N
      IK=(K-1)*N+I
  100 B(I)=B(I)+A(IK)*BK
      NN=N*N
      B(N)=B(N)/A(NN)
      DO 300 K=2,N
      I=NP1-K
      J1=I+1
      BI=B(I)
      DO 200 J=J1,N
      IJ=(J-1)*N+I
  200 BI=BI-A(IJ)*B(J)
      B(I)=BI
      II=(I-1)*N+I
  300 B(I)=B(I)/A(II)
      RETURN
      END
```

```
      SUBROUTINE FNAME(JOBNAME,EXT,FN)
C
C*******************************************************************
C                                                                  *
C    THIS SUBROUTINE TAKES AN ALPHANUMERIC JOBNAME CONTAINED IN    *
C    THE ARRAY JOBNAME(2) AND CONCATENATES IT WITH THE EXTENSION   *
C    NAME CONTAINED IN EXT AND RETURNS THE COMPOSED FILE NAME WITH *
C    A PERIOD SEPARATING THE FILE NAME AND FILE EXTENSION. THE     *
C    COMPLETE NAME IS RETURNED LEFT JUSTIFIED IN THE ARRAY FN(3)   *
C    THIS  VERSION TAKES ADVANTAGE OF FORTRAN-77 AND SHOULD BE     *
C    MACHINE INDEPENDENT. IT HAS WORKED ON THE VAX/780 THE         *
C    APOLLO MODEL DN/300, AND THE IBM PC.                          *
C                                                                  *
C*******************************************************************
C                                                                  *
C                COPYRIGHT (C) BY GILLES CANTIN                    *
C              MONTEREY, CALIFORNIA,  24 JULY 1984.                *
C                                                                  *
C*******************************************************************
C
      DIMENSION JOBNAME(2),FN(3),JJOB(2),FFN(3)
      CHARACTER*1 JOBCH(8),FNCH(12),EXTCH(4),BLANK,PERIOD
      EQUIVALENCE (JJOB(1),JOBCH(1)),(EEXT,EXTCH(1)),(FFN(1),FNCH(1))
      DATA BLANK/' '/,PERIOD/'.'/
      DO 10 I=1,4
 10   EXTCH(I)=BLANK
      DO 20 I=1,8
 20   JOBCH(I)=BLANK
      DO 30 I=1,12
 30   FNCH(I)=BLANK
      DO 40 I=1,2
 40   JJOB(I)=JOBNAME(I)
      EEXT=EXT
      EXTCH(4)=EXTCH(3)
      EXTCH(3)=EXTCH(2)
      EXTCH(2)=EXTCH(1)
      EXTCH(1)=PERIOD
      DO 50 I=1,8
      IF(JOBCH(I).NE.BLANK) II=I
 50   FNCH(I)=JOBCH(I)
      IL=II+1
      IH=IL+3
      DO 60 I=IL,IH
      III=I-II
 60   FNCH(I)=EXTCH(III)
      DO 70 I=1,3
 70   FN(I)=FFN(I)
      RETURN
      END
```

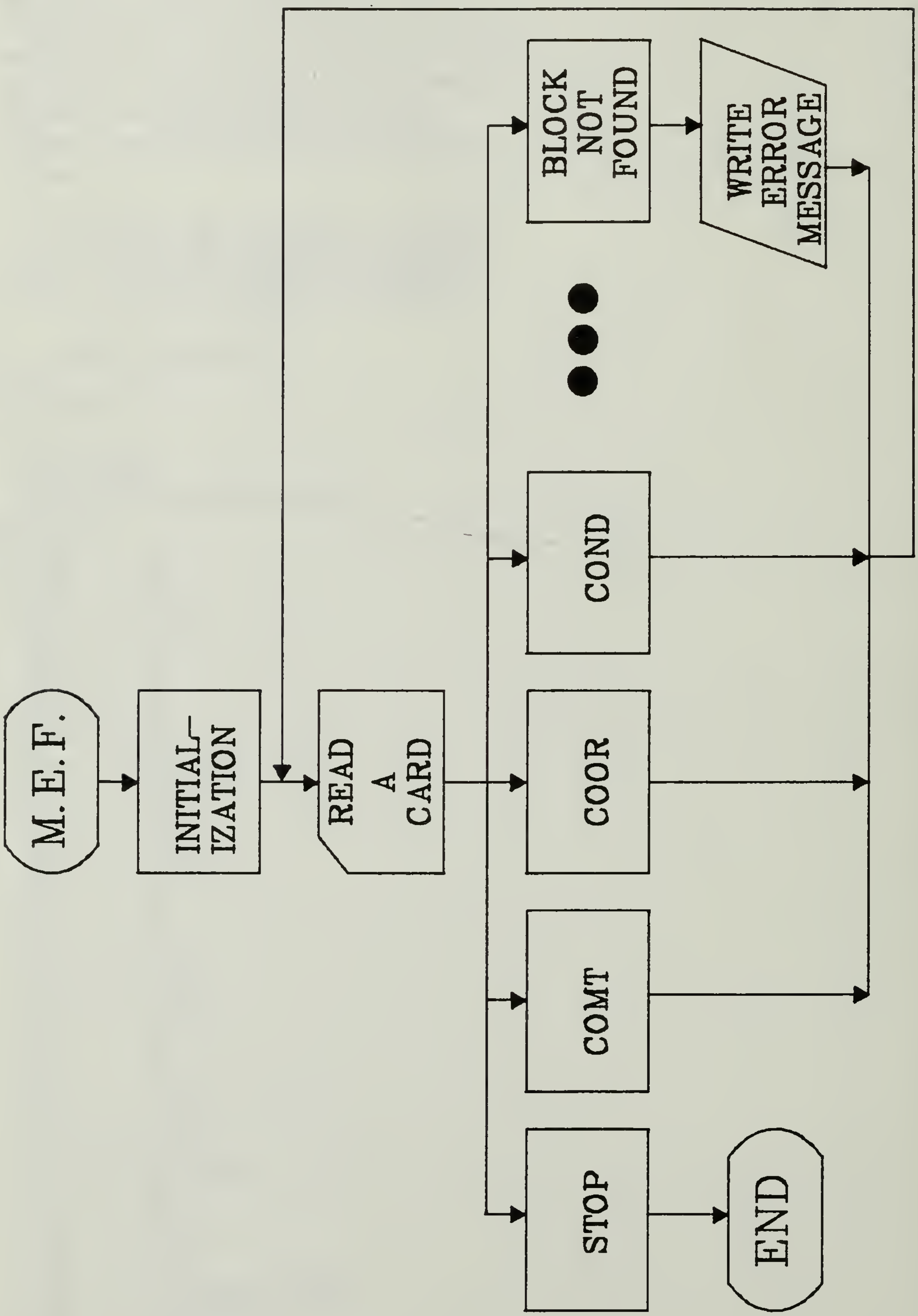
```
;       SUBROUTINE TICKER(ITIME)
;
;       THIS IS AN 8088 ASSEMBLY LANGUAGE ROUTINE
;       ADAPTED FROM A PROGRAM WRITTEN BY W. CLAFF OF THE
;       BOSTON COMPUTER SOCIETY, ONE CENTER PLAZA, BOSTON,
;       MASS 02108.  THE ORIGINAL VERSION WAS FOR MICROSOFT
;       FORTRAN V3.1, AND WAS PUBLISHED IN BYTE MAGAZINE,
;       FEB 1984.  THIS VERSION HAS BEEN MODIFIED TO
;       CONFORM TO THE CALLING CONVENTION FOR MICROSOFT
;       FORTRAN V3.2.
;
;       AS IN THE VERSION WRITTEN BY CLAFF, THIS VERSION
;       EXTRACTS THE BCD TIME FROM DOS, AND RETURNS THE
;       RESULT TO THE CALLING PROGRAM IN CENTISECONDS.
;
DATA        SEGMENT PUBLIC 'DATA'
DATA    ENDS
DGROUP  GROUP DATA

CODE    SEGMENT 'CODE'
        ASSUME CS:CODE,DS:DGROUP,SS:DGROUP
PUBLIC  TICKER
TICKER  PROC        FAR
        PUSH        BP
        MOV         BP,SP
        PUSH        AX
        PUSH        BX
        PUSH        CX
        PUSH        DX
        MOV         AH, 02CH
        INT         021H
        XCHG        CX,DX
        MOV         AL, CH
        MOV         BL, 100
        MUL         BL
        MOV         CH, 0
        ADD         CX, AX
        MOV         AL, DH
        MOV         BL, 60
        MUL         BL
        MOV         DH, 0
        ADD         AX, DX
        MOV         DX, 0
        MOV         BX, 6000
        MUL         BX
        ADD         CX, AX
        ADC         DX, 0
        LES         BX, DWORD PTR [BP+6]
        MOV         ES:[BX],CX
        MOV         ES:[BX+2],DX
```

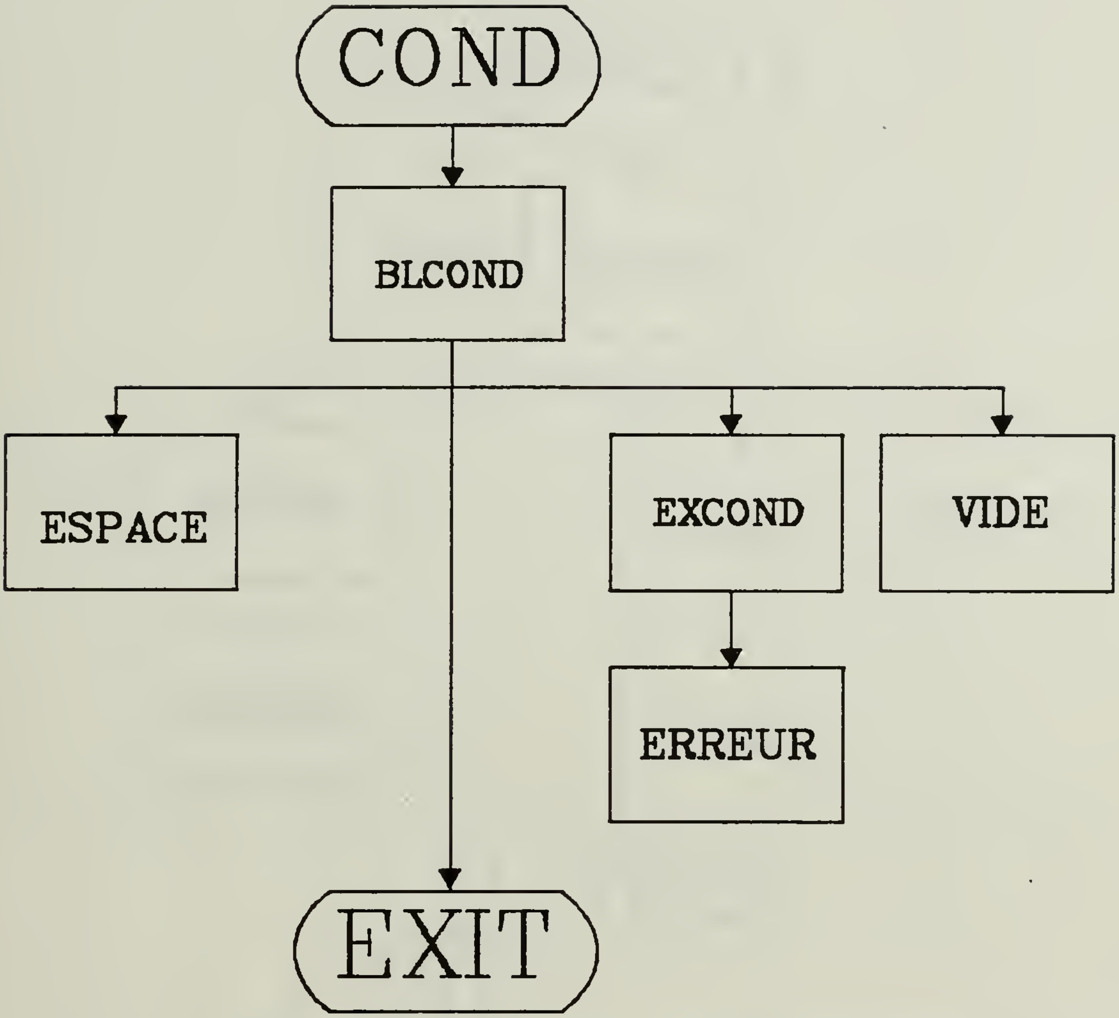
```
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        POP     BP
        RET     4
TICKER  ENDP
CODE    ENDS
        END
```

APPENDIX C

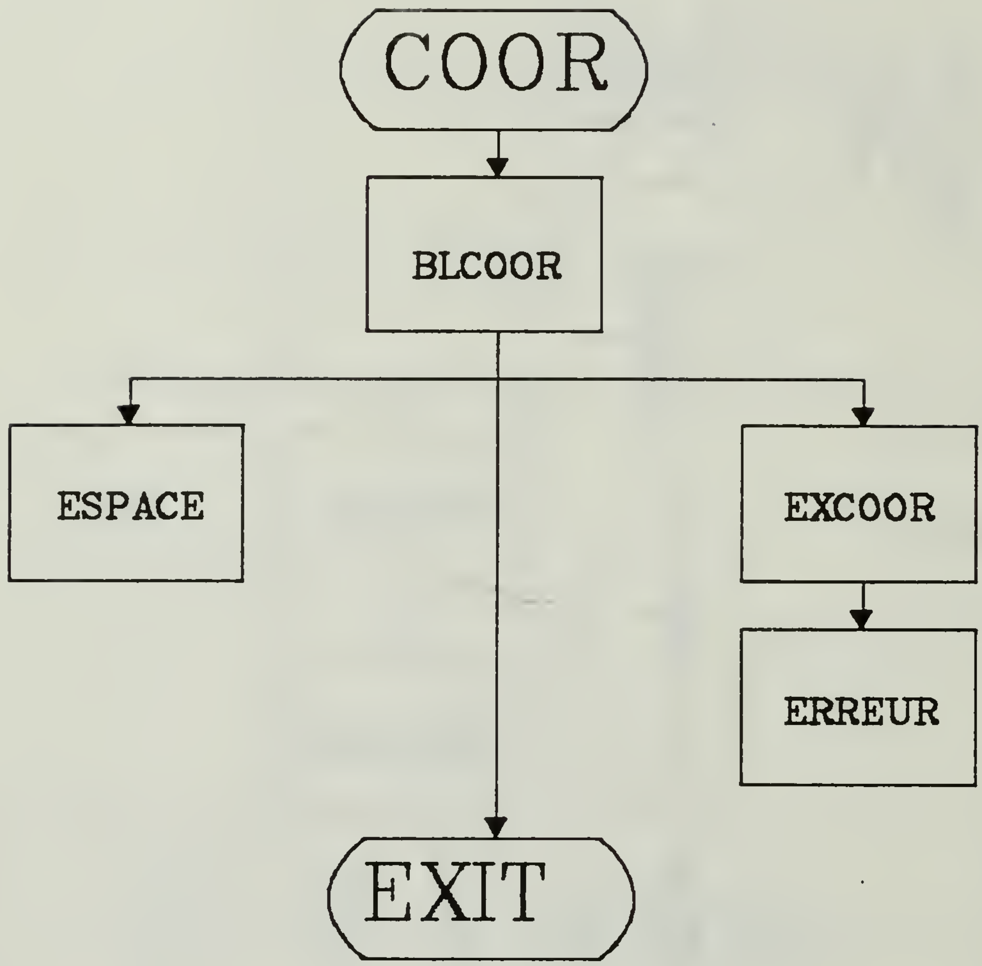
FUNCTIONAL BLOCK DIAGRAMS



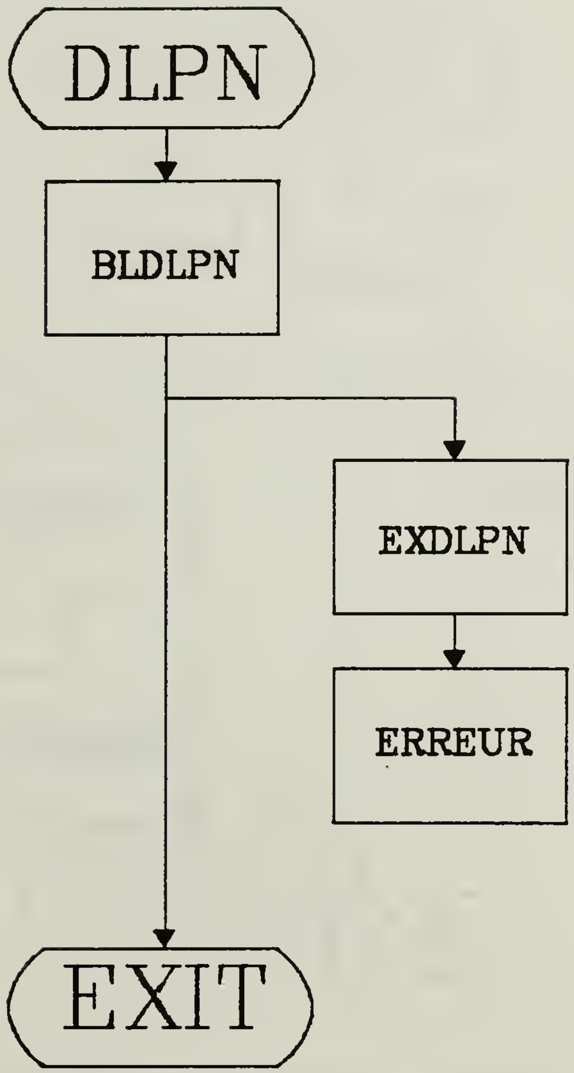
OVER-ALL FUNCTIONAL BLOCK DIAGRAM



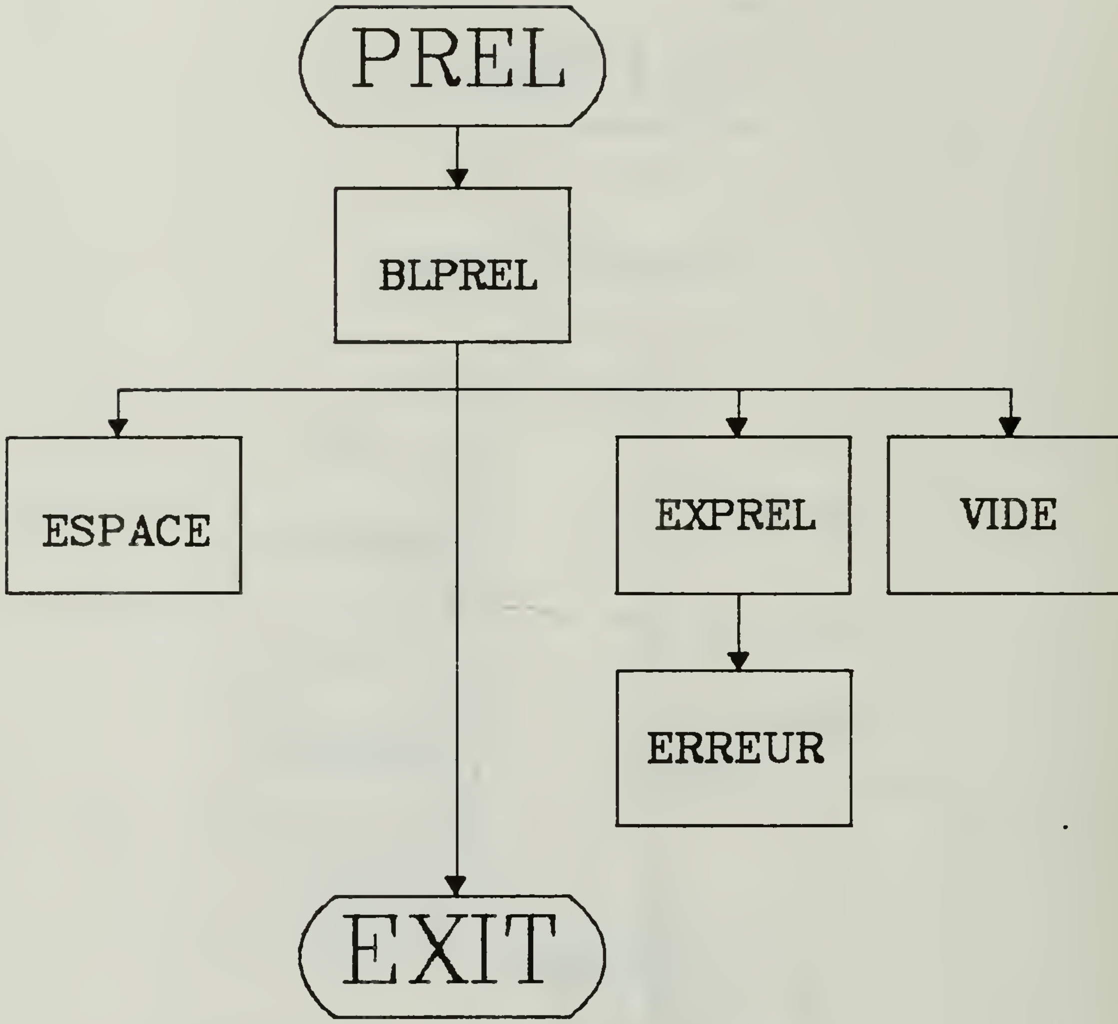
FUNCTIONAL BLOCK COND



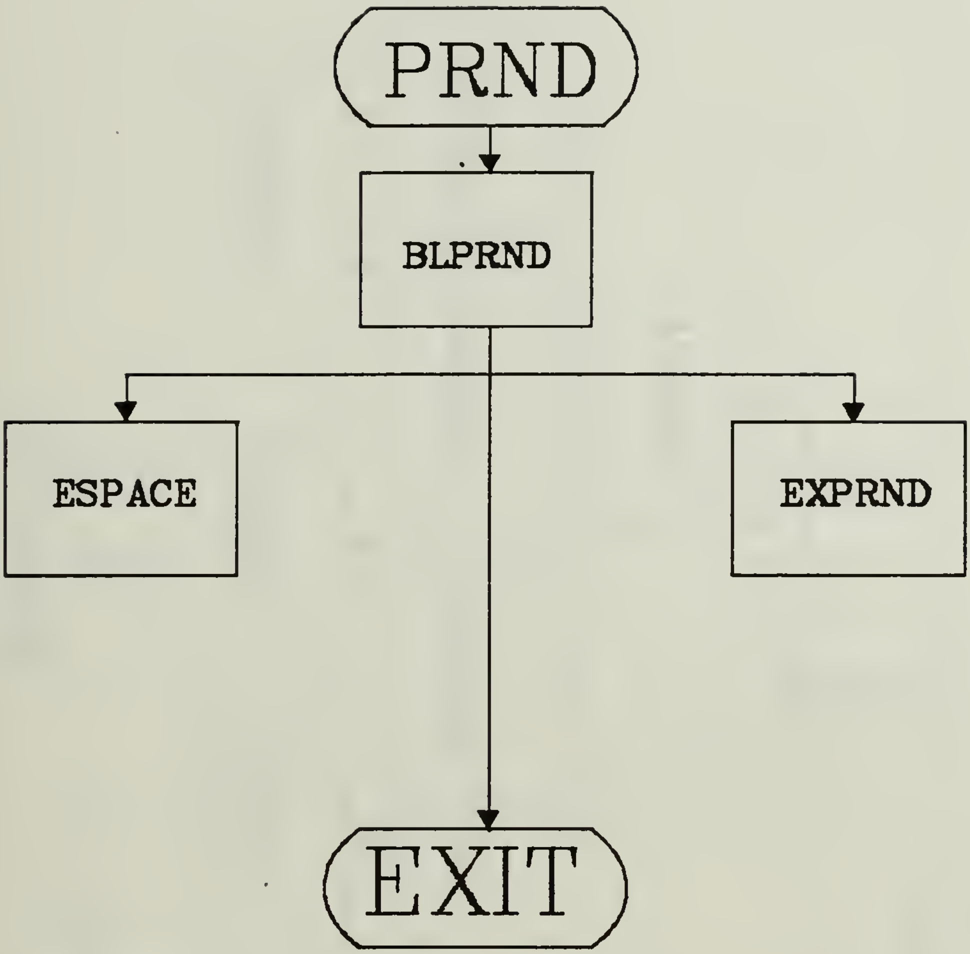
FUNCTIONAL BLOCK COOR



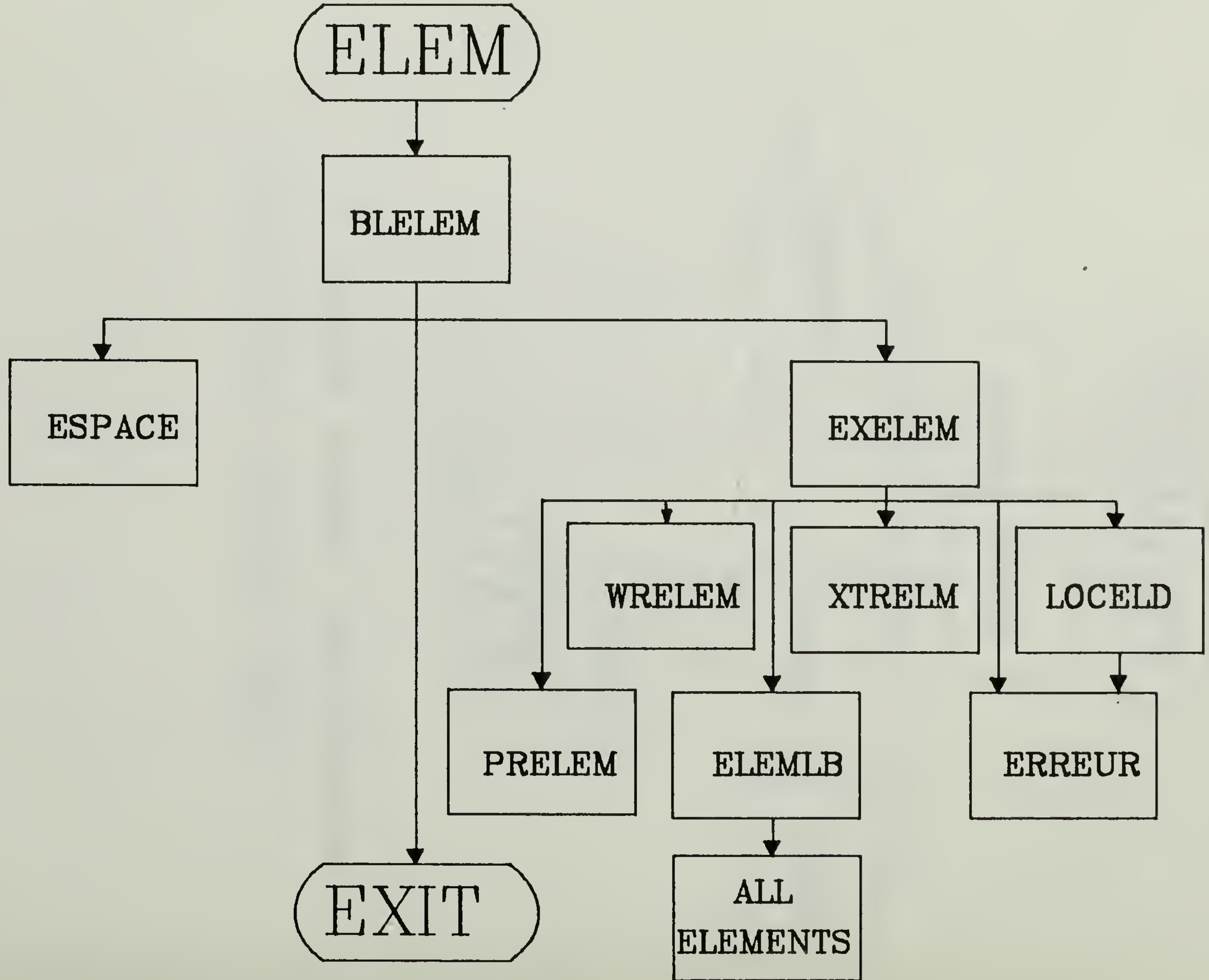
FUNCTIONAL BLOCK DLPN



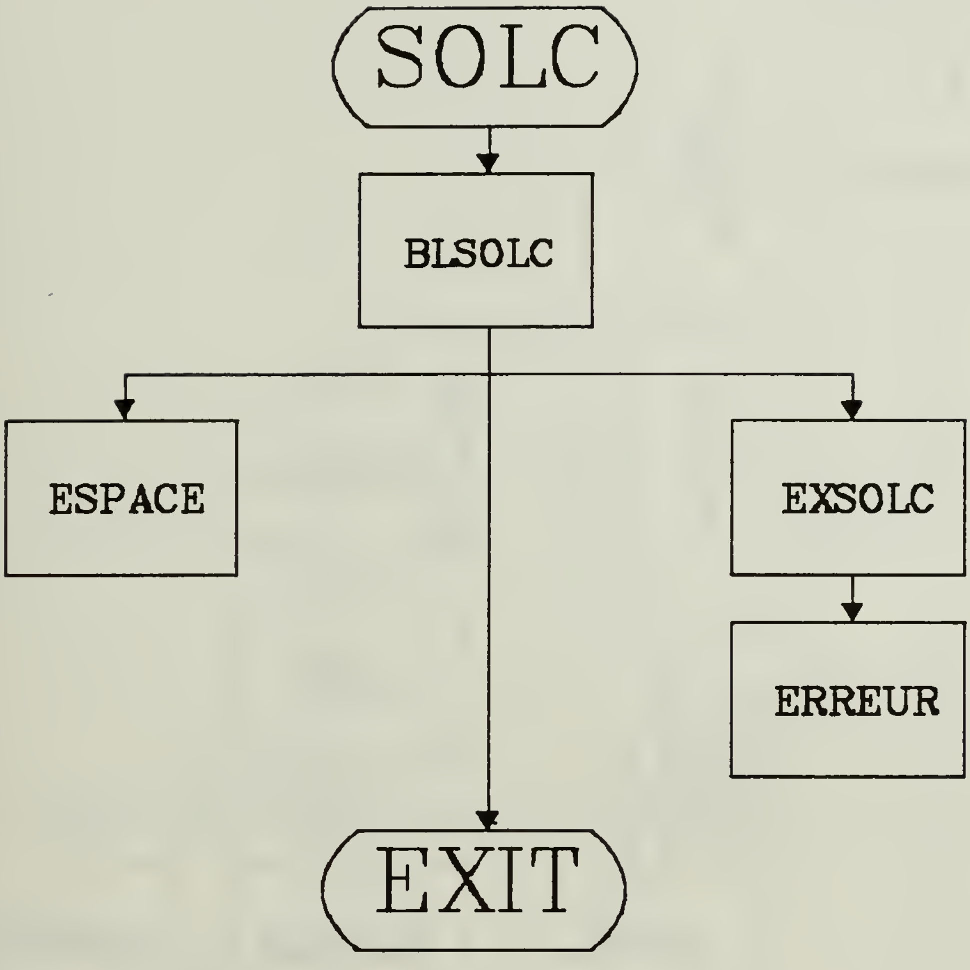
FUNCTIONAL BLOCK PREL



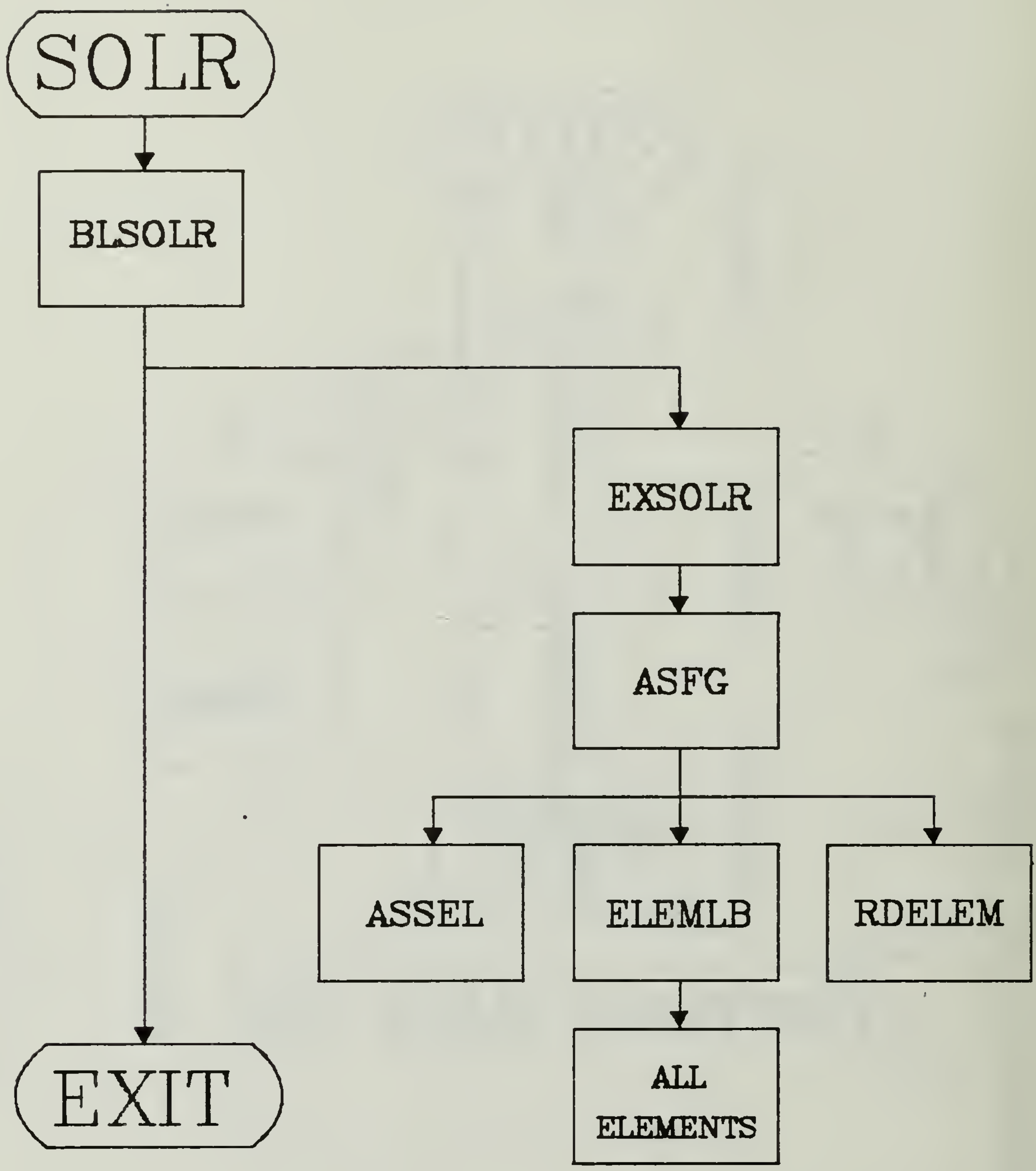
FUNCTIONAL BLOCK PRND



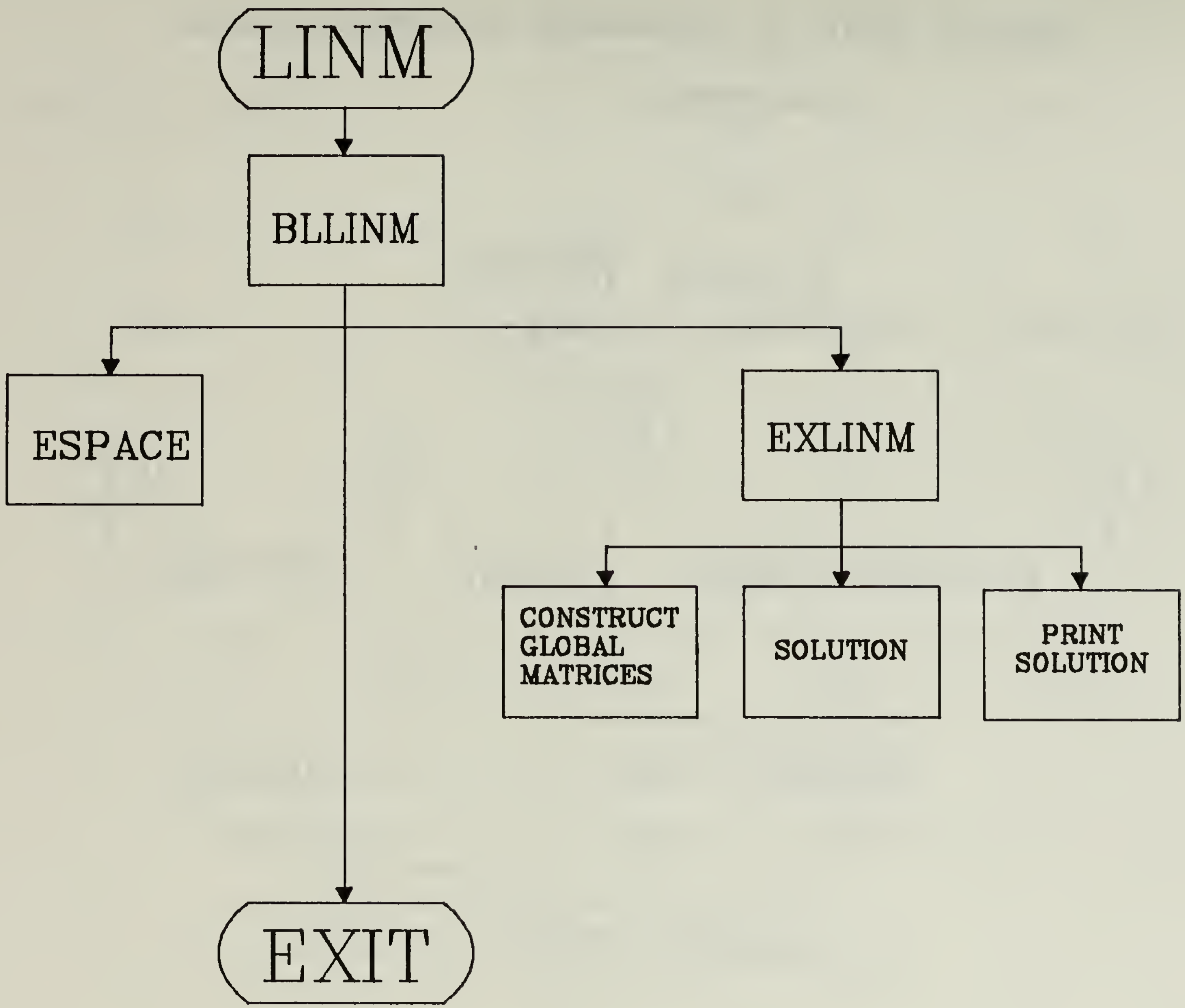
FUNCTIONAL BLOCK ELEM



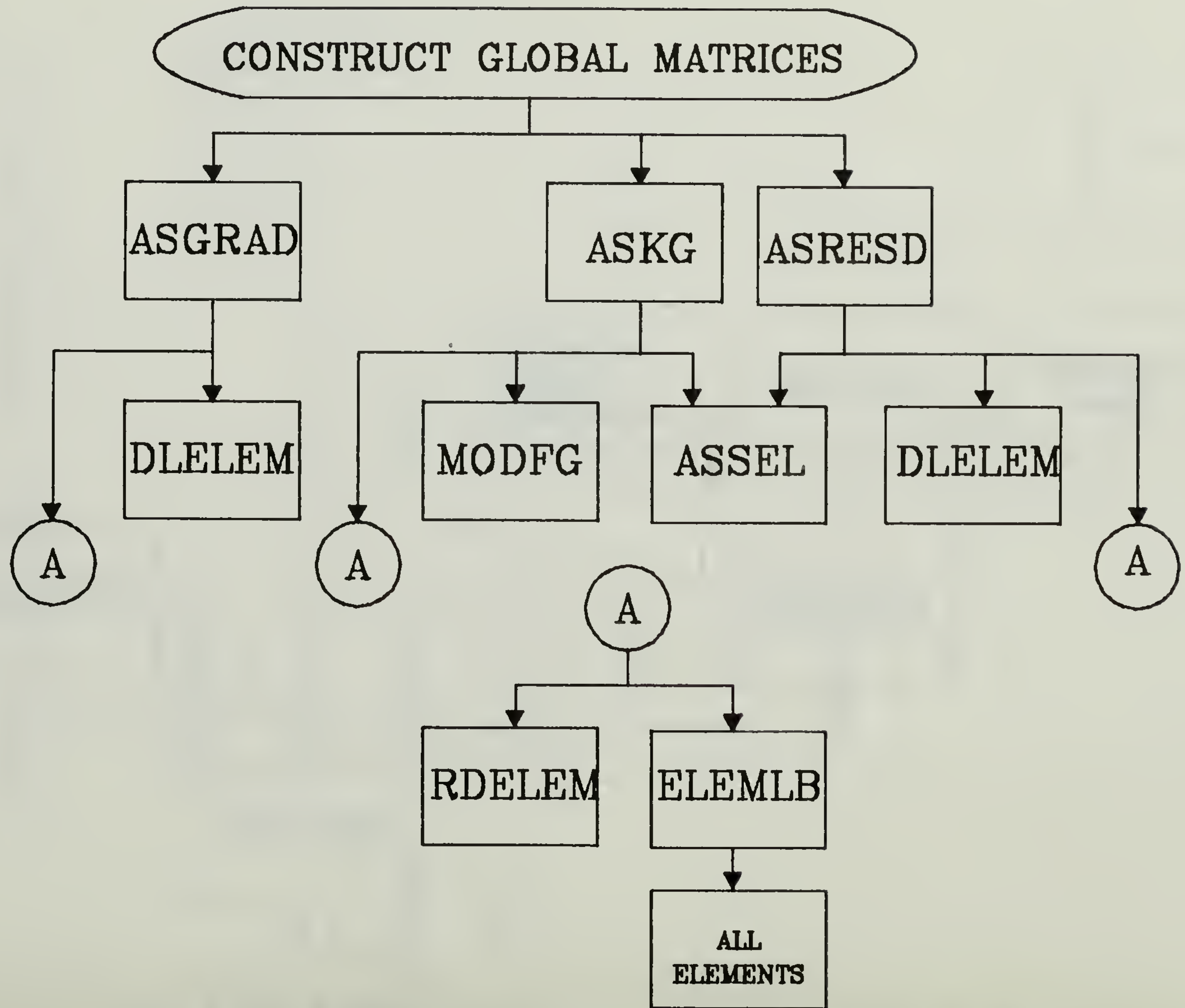
FUNCTIONAL BLOCK SOLC



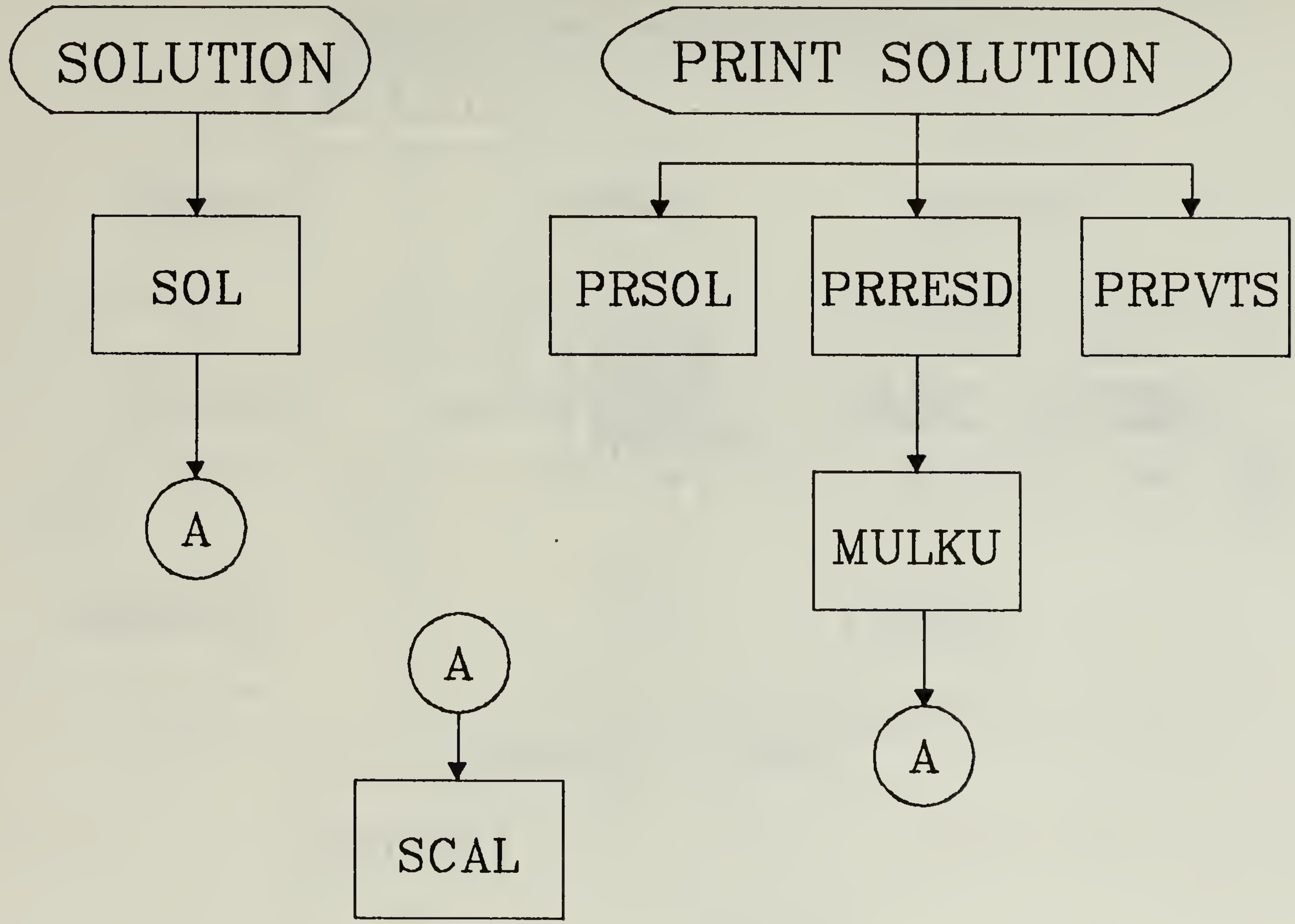
FUNCTIONAL BLOCK SOLR



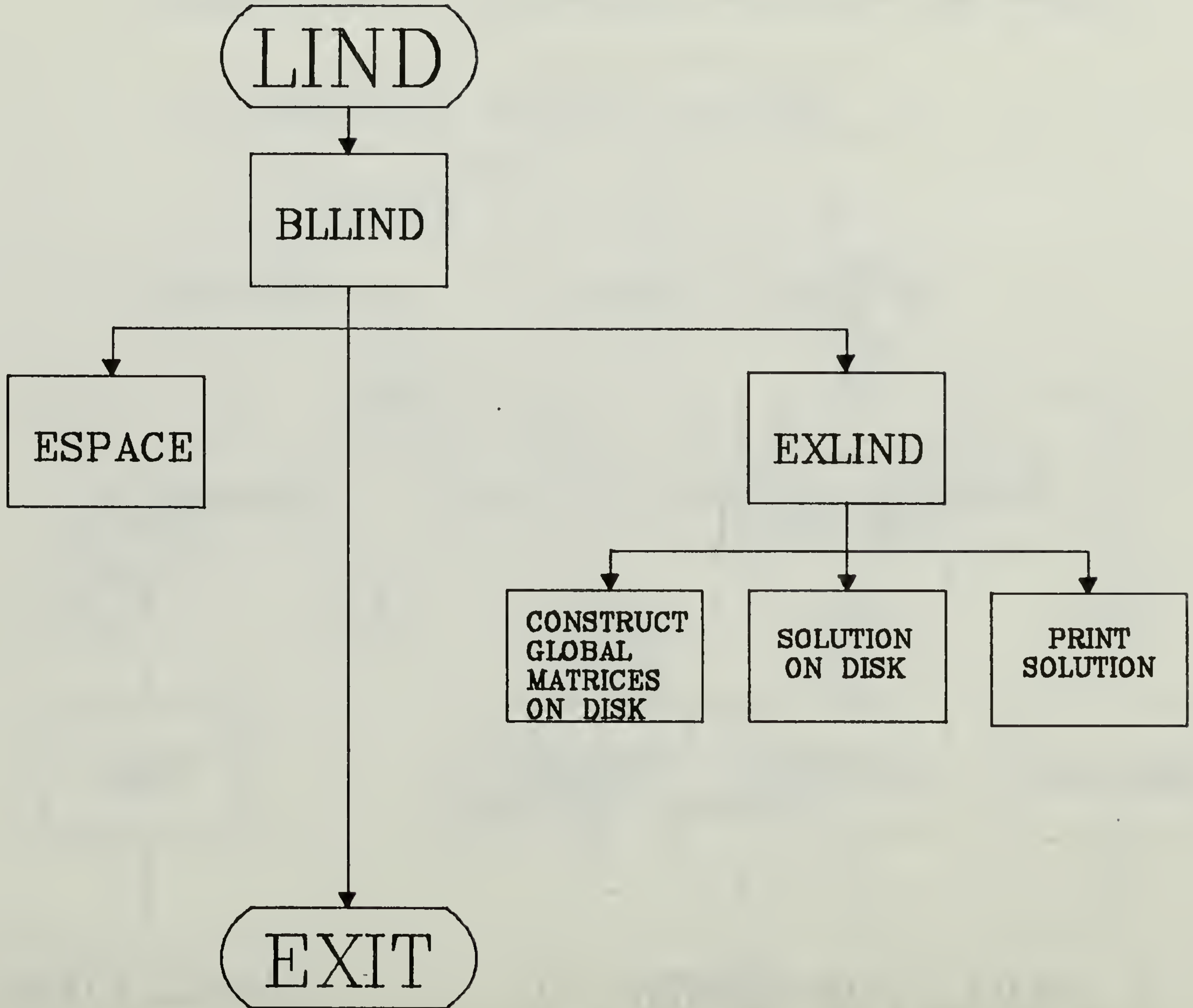
FUNCTIONAL BLOCK LINM



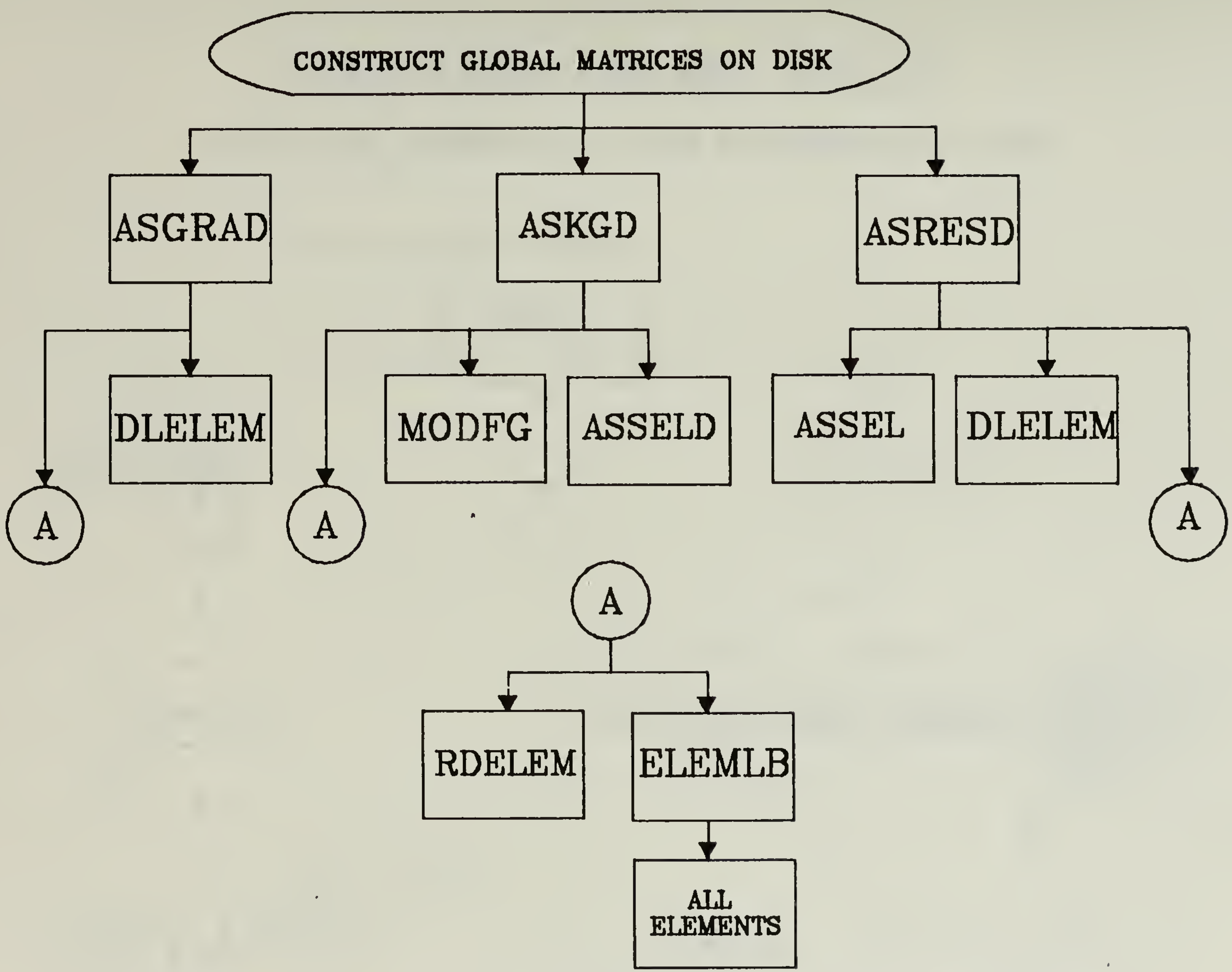
FUNCTION COMMON TO CONSTRUCTION OF GLOBAL MATRICES



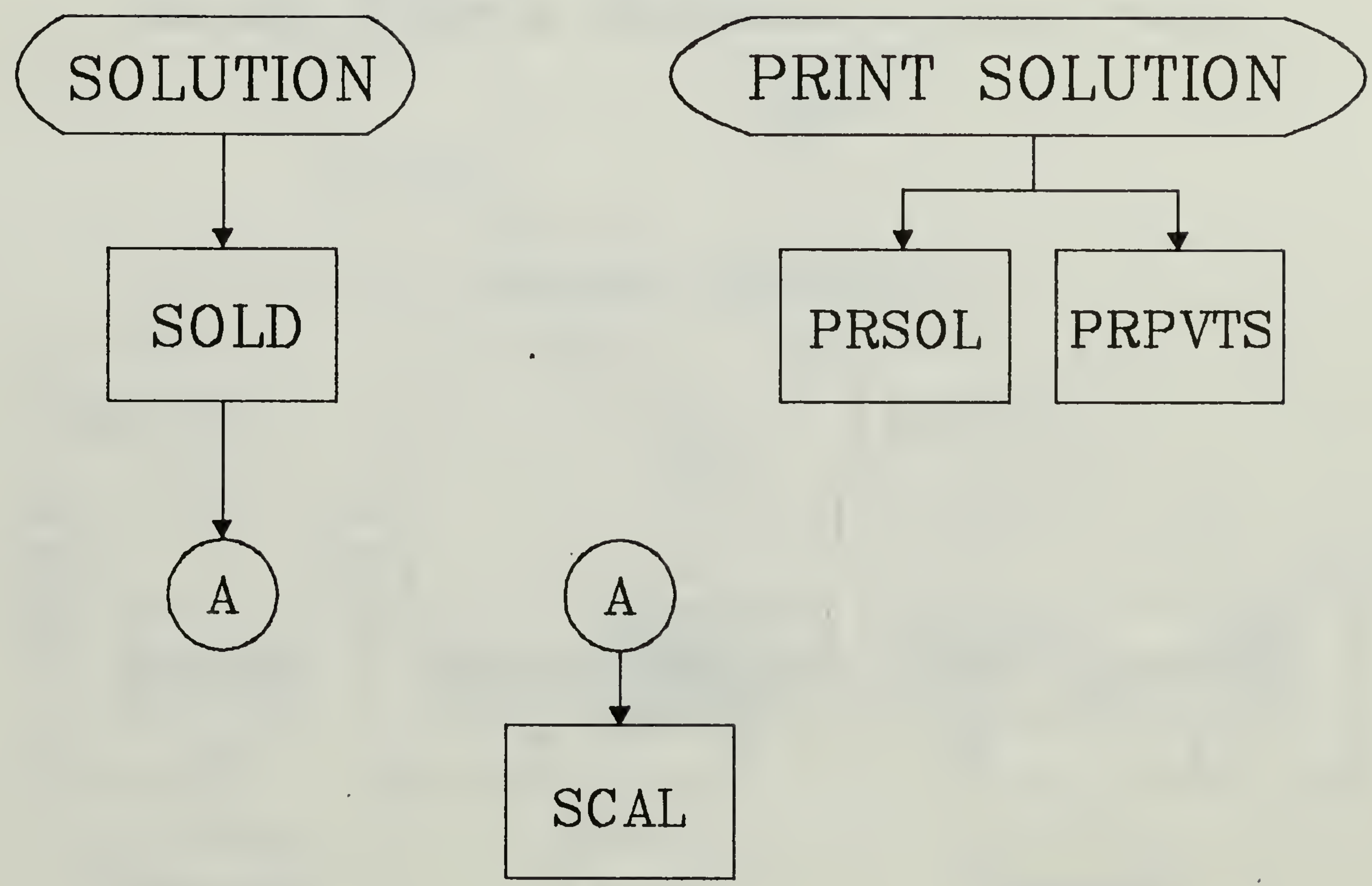
FUNCTIONS COMMON TO THE SOLUTION AND PRINTING THE SOLUTION



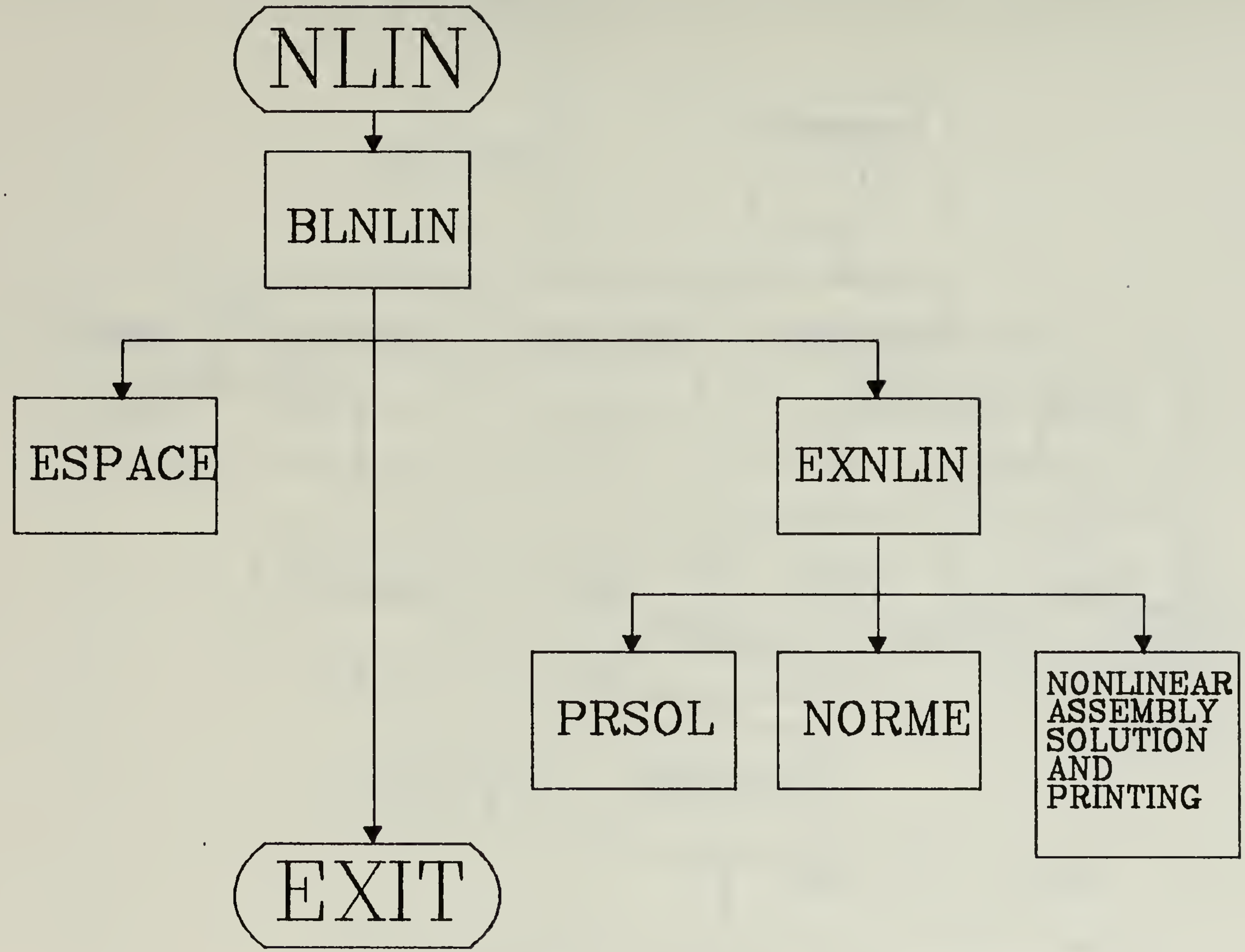
FUNCTIONAL BLOCK LIND



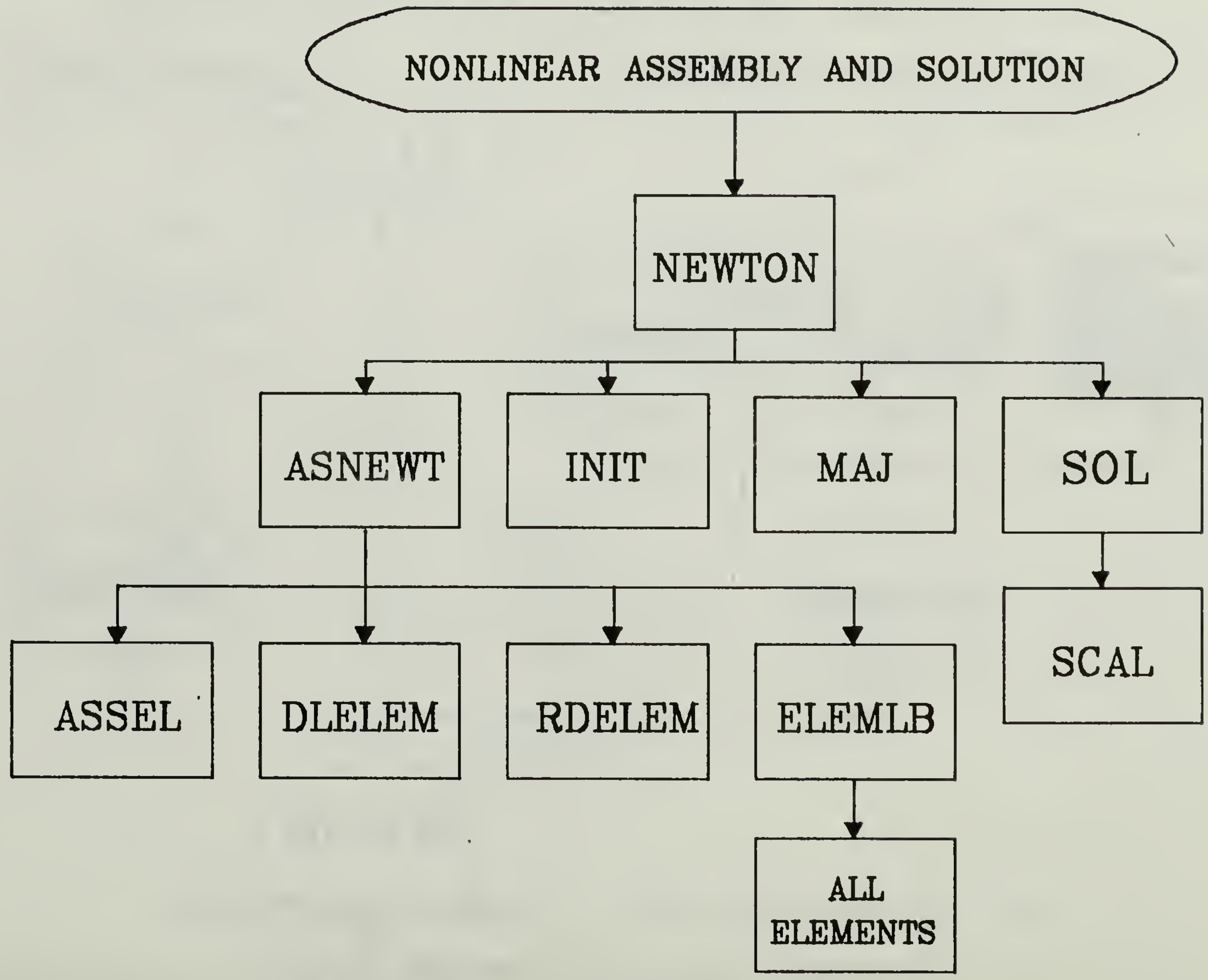
FUNCTIONS COMMON TO CONSTRUCTION OF GLOBAL MATRICES ON DISK



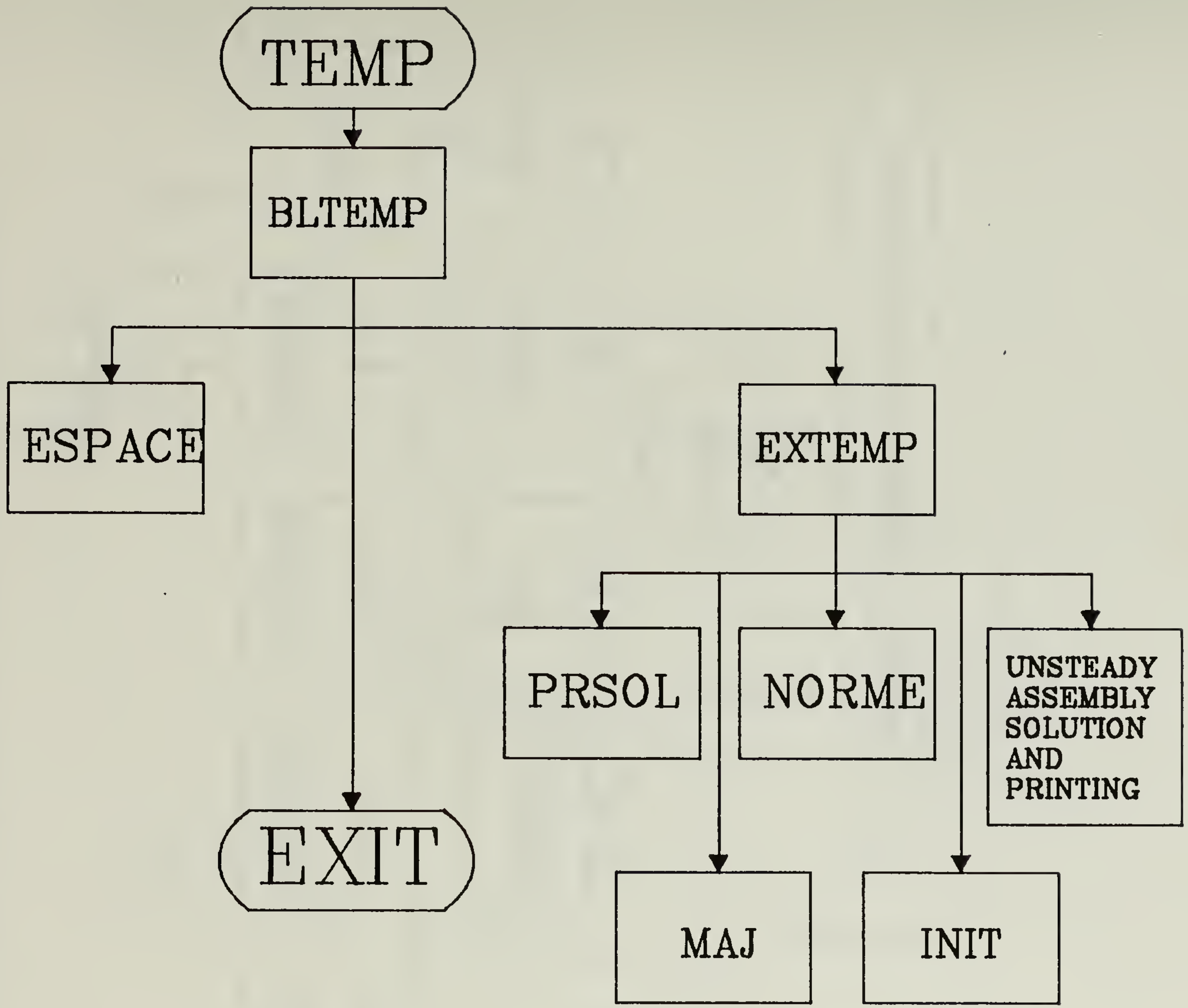
FUNCTIONS COMMON TO THE SOLUTION ON DISK
AND PRINTING THE DISK SOLUTION



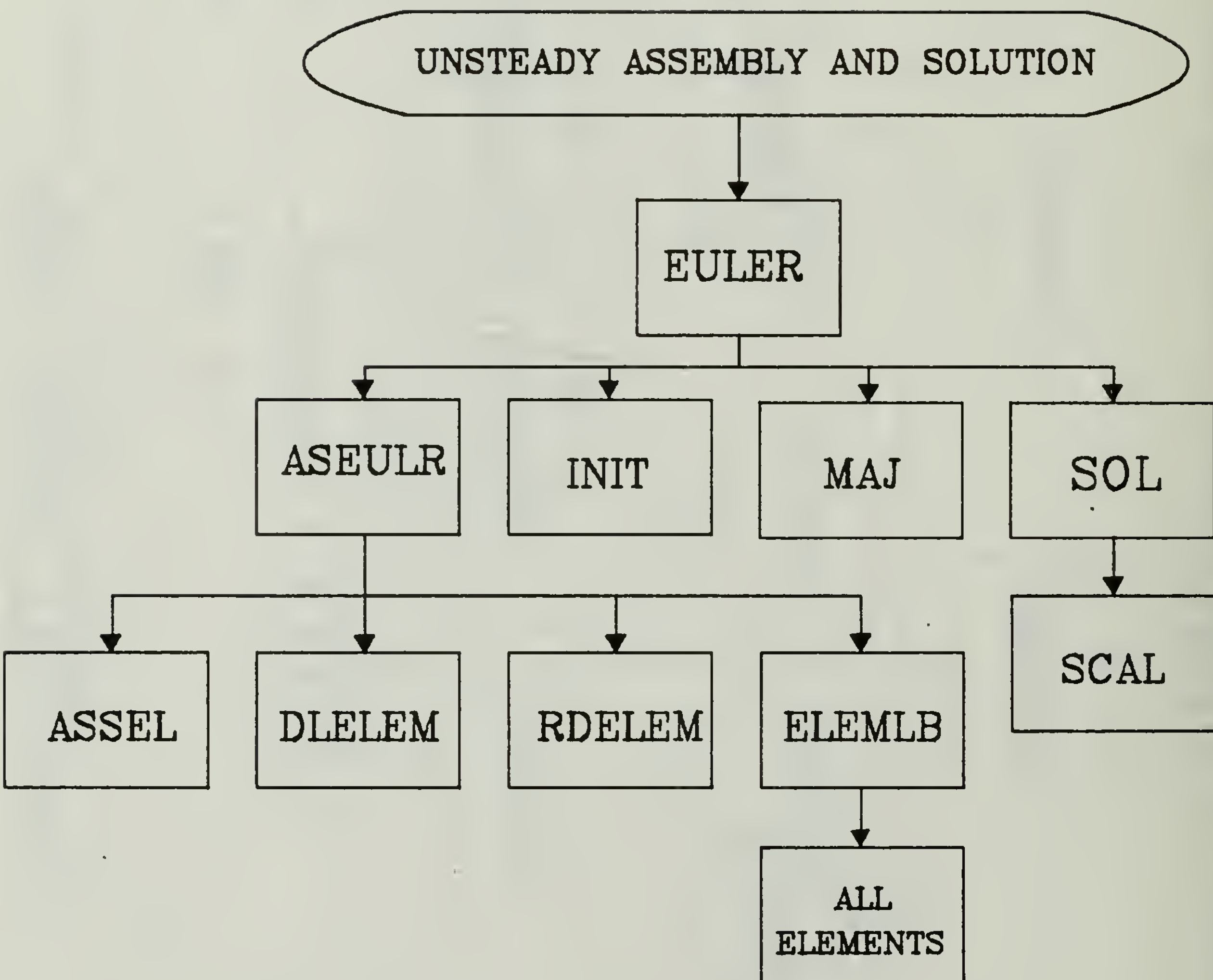
FUNCTIONAL BLOCK NLIN



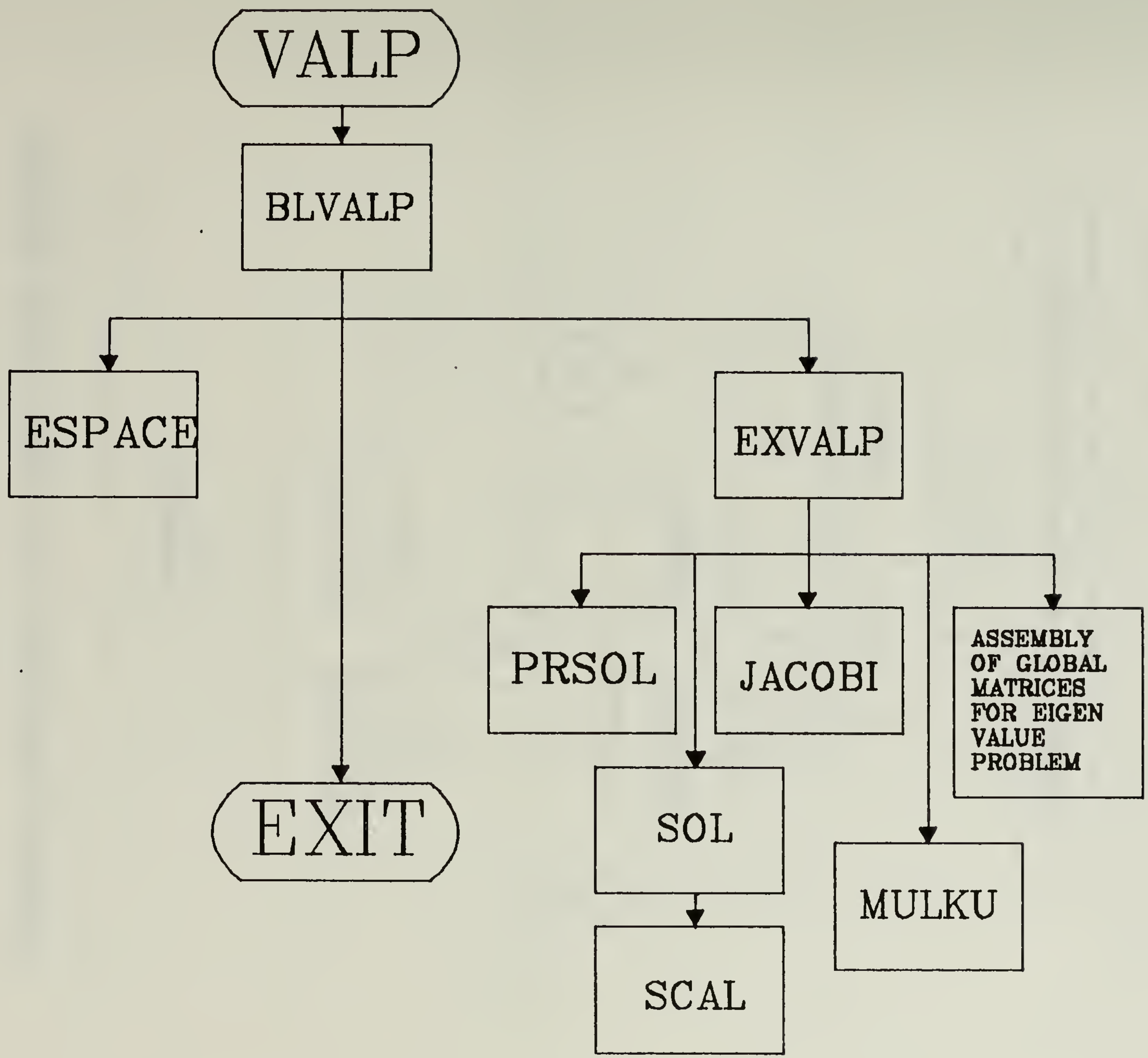
FUNCTIONS FOR NONLINEAR ASSEMBLY, SOLUTION, AND PRINTING



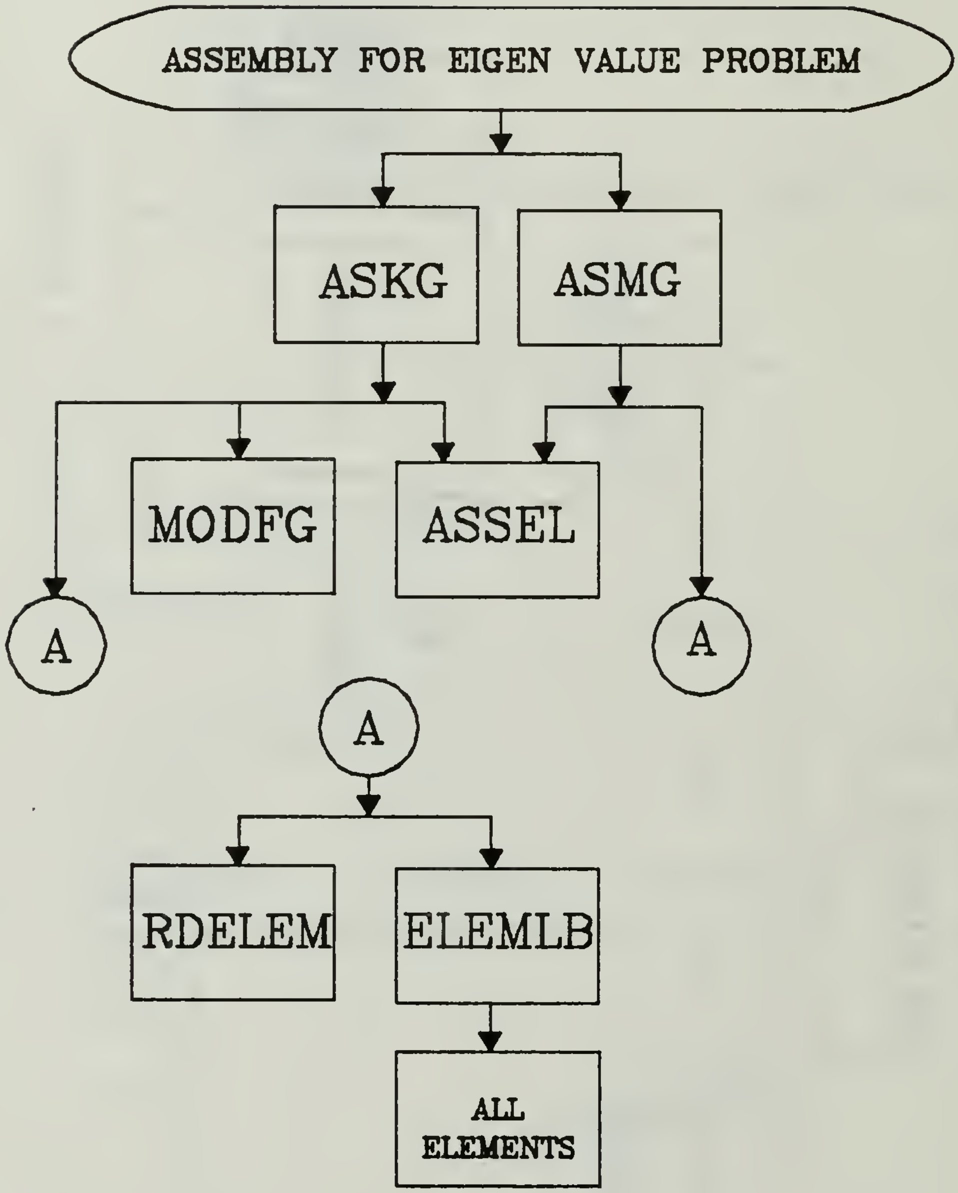
FUNCTIONAL BLOCK TEMP



FUNCTIONS FOR UNSTEADY ASSEMBLY, SOLUTION AND PRINTING



FUNCTIONAL BLOCK VALP



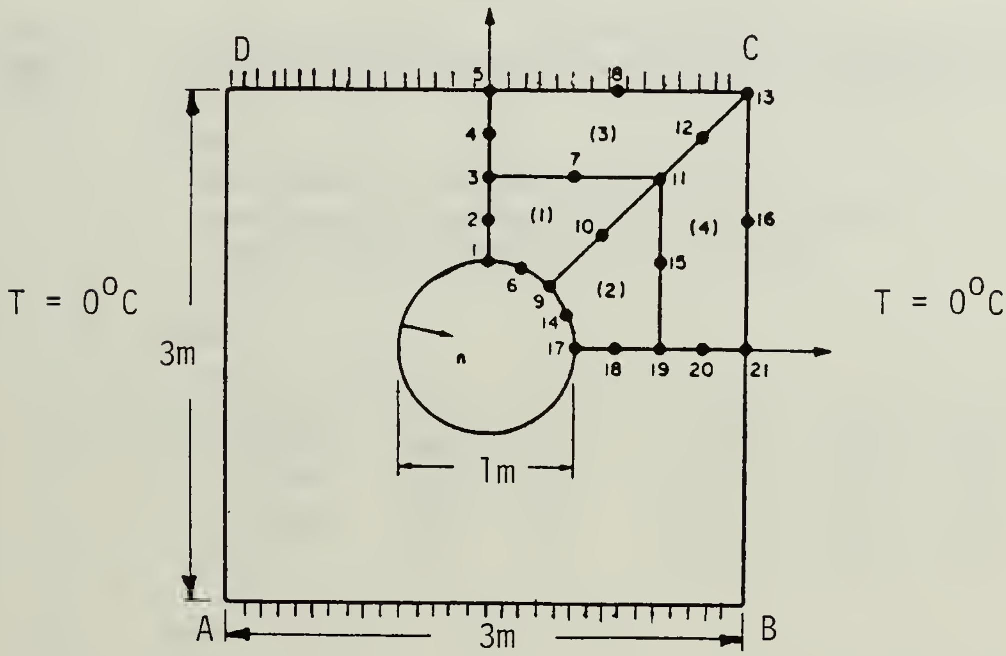
ASSEMBLY FUNCTIONS FOR EIGEN VALUE PROBLEM

APPENDIX D

SAMPLE PROBLEMS AND SOLUTIONS

Conduction heat transfer problem for comparison with the results of Dhatt and Touzot

concrete plate



$$d = d_x = d_y = 1.4 \text{ w/(m } ^oC)$$

$$C = 2.03 \times 10^6 \text{ J/(m}^3\ ^oC)$$

constant heat flux on inside = 1

The distributed boundary condition on the inner circle is replaced by consistent concentrated nodal values:

nodes 1, 17 = 0.6545
nodes 6, 14 = 0.2618
node 9 = 0.1309

The consistent nodal values are arrived at as follows:

$$0.6545 = \frac{\pi}{24}$$

$$0.2618 = \frac{\pi}{6}$$

$$0.1309 = \frac{\pi}{12}$$

In the analysis, the double symmetry allows only one quarter of the plate to be considered.

F.E.M.3.
G.TOUZOT, G.DHATT
MODIFIED BY

REHE E. RUESCH

=================

IMAGE OF DATA CARDS

===========================

```
                                      C O L U M N   N U M B E R
  CARD                 1         2         3         4         5         6         7         8
 NUMBER       12345678901234567890123456789012345678901234567890123456789012345678901234567890
 -------      --------------------------------------------------------------------------------
       1      COMT
       2        HEAT TRANSFER IN A PERFORATED SQUARE PLATE
       3        SAMPLE PROBLEM TO COMPARE RESULTS OF MEF ON THE IBM PC
       4           WITH THOSE OF THE AUTHORS, DHATT AND TOUZOT
       5
       6      COOR
       7         21    1    2   0.5       0.5
       8          1 0.0        1.0       0.0           5 0.0       3.0       0.0
       9          6 0.3827     0.9239    0.0           8 1.5       3.0       0.0
      10          9 0.707      0.707     0.0          13 3.0       3.0       0.0
      11         14 0.9239     0.3827    0.0          16 3.0       1.5       0.0
      12         17 1.0        0.0       0.0          21 3.0       0.0       0.0
      13          0
      14      COND
      15      1
      16         13   16   21
      17      0
      18      PREL
      19          1    4
      20          1 1.4       1.4       1.4            2.03E6
      21      0
      22      ELEM
      23          4    8    1
      24          1    2    8    1    1    1    1    6    9   10   11    7    3    2
      25          3    2    8    1    1    1    3    7   11   12   13    8    5    4
      26          0
      27      SOLC     3
      28          1 0.06545
      29          1   17
      30          2 0.1309
      31          9
```

```
   32          3 0.2618
   33          6   14
   34          0
   35        LINM
   36          1
   37        STOP
 ------      ---------------------------------------------------------------------------------
 CARD        12345678901234567890123456789012345678901234567890123456789012345678901234567890
 NUMBER               1         2         3         4         5         6         7         8
                                     C O L U M N   N U M B E R
```

E N D   O F   D A T A

COMMENTS
==========

HEAT TRANSFER IN A PERFORATED SQUARE PLATE
SAMPLE PROBLEM TO COMPARE RESULTS OF MEF ON THE IBM PC
WITH THOSE OF THE AUTHORS, DHATT AND TOUZOT

INPUT OF NODES (M= 0)
==================

```
MAX. NUMBER OF NODES                (NNT)=   21
MAX. NUMBER OF D.O.F. PER NODE     (NDLN)=    1
DIMENSIONS OF THE PROBLEM          (NDIM)=    2
COORDINATE SCALE FACTORS            (FAC)=  .50000E+00  .50000E+00  .10000E+01
WORKSPACE IN REAL WORDS             (NVA)=     20000
```

INPUT OF BOUNDARY CONDITIONS (M= 0)
==================================

BOUNDARY CONDITIONS CARDS

```
>>>>>1000000000  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00
>>>>>              13   16   21    0    0    0    0    0    0    0    0    0    0    0    0    0
>>>>>0000000000  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00
```

```
TOTAL NUMBER OF NODES                        (NNT)=   21
TOTAL NUMBER OF D.O.F.                      (NDLT)=   21
NUMBER OF EQUATIONS TO BE SOLVED             (NEQ)=   18
NUMBER OF PRESCRIBED NON ZERO D.O.F.       (NCLNZ)=    0
NUMBER OF PRESCRIBED ZERO D.O.F.            (NCLZ)=    3
```

TOTAL NUMBER OF PRESCRIBED D.O.F. (NCLT)= 3

NODAL COORDINATES ARRAY

NO	D.L.	X	Y	Z	EQUATION NUMBER (NEQ)
1	1	.00000E+00	.50000E+00	.00000E+00	1
2	1	.00000E+00	.75000E+00	.00000E+00	2
3	1	.00000E+00	.10000E+01	.00000E+00	3
4	1	.00000E+00	.12500E+01	.00000E+00	4
5	1	.00000E+00	.15000E+01	.00000E+00	5
6	1	.19135E+00	.46195E+00	.00000E+00	6
7	1	.47068E+00	.98097E+00	.00000E+00	7
8	1	.75000E+00	.15000E+01	.00000E+00	8
9	1	.35350E+00	.35350E+00	.00000E+00	9
10	1	.64013E+00	.64013E+00	.00000E+00	10
11	1	.92675E+00	.92675E+00	.00000E+00	11
12	1	.12134E+01	.12134E+01	.00000E+00	12
13	1	.15000E+01	.15000E+01	.00000E+00	-1
14	1	.46195E+00	.19135E+00	.00000E+00	13
15	1	.98097E+00	.47068E+00	.00000E+00	14
16	1	.15000E+01	.75000E+00	.00000E+00	-2
17	1	.50000E+00	.00000E+00	.00000E+00	15
18	1	.75000E+00	.00000E+00	.00000E+00	16
19	1	.10000E+01	.00000E+00	.00000E+00	17
20	1	.12500E+01	.00000E+00	.00000E+00	18
21	1	.15000E+01	.00000E+00	.00000E+00	-3

INPUT OF ELEMENT PROPERTIES (M= 0)
==================================

NUMBER OF GROUPS OF PROPERTIES (NGPE)= 1
NUMBER OF PROPERTIES PER GROUP (NPRE)= 4

CARDS OF ELEMENT PROPERTIES

)))))	1	.14000E+01	.14000E+01	.14000E+01	.20300E+07
)))))	0	.00000E+00	.00000E+00	.00000E+00	.00000E+00

INPUT OF ELEMENTS (M= 0)
===================

MAX. NUMBER OF ELEMENTS (NELT)= 4
MAX. NUMBER OF NODES PER ELEMENT (NNEL)= 8
DEFAULT ELEMENT TYPE (NTPE)= 1
NUMBER OF GROUPS OF ELEMENTS (NGRE)= 1
INDEX FOR NON SYMMETRIC PROBLEM (NSYM)= 0
INDEX FOR IDENTICAL ELEMENTS (NIDENT)= 0

```
ELEMENT:    1 TYPE: 1 N.P.: 8 D.O.F.:  8 N. PROP:  0 EL. PROP:  4 GROUP:  1
    CONNECTIVITY (NE)    1    6    9   10   11    7    3    2
ELEMENT:    2 TYPE: 1 N.P.: 8 D.O.F.:  8 N. PROP:  0 EL. PROP:  4 GROUP:  1
    CONNECTIVITY (NE)    9   14   17   18   19   15   11   10
ELEMENT:    3 TYPE: 1 N.P.: 8 D.O.F.:  8 N. PROP:  0 EL. PROP:  4 GROUP:  1
    CONNECTIVITY (NE)    3    7   11   12   13    8    5    4
ELEMENT:    4 TYPE: 1 N.P.: 8 D.O.F.:  8 N. PROP:  0 EL. PROP:  4 GROUP:  1
    CONNECTIVITY (NE)   11   15   19   20   21   16   13   12

    MEAN BAND HEIGHT=      5.3 MAXIMUM=   10
    LENGTH OF A TRIANGLE IN KG         (NKG)=        95
    NUMBER OF INTEGRATION POINTS       (NPG)=        36


INPUT OF CONCENTRADED LOADS (M= 3)
===================================
                                              TABLE FG   GOES FROM VA(    111) TO VA(    128)


CARDS OF NODAL LOADS


>>>>>    1  .65450E-01
>>>>>    1   17    0    0    0    0    0    0    0    0    0    0    0    0    0    0
>>>>>    2  .13090E+00
>>>>>    9    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
>>>>>    3  .26180E+00
>>>>>    6   14    0    0    0    0    0    0    0    0    0    0    0    0    0    0
>>>>>    0  .00000E+00


TOTAL LOAD VECTOR
          .65450E-01  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .26180E+00  .00000E+00  .00000E+00  .1309
00  .00000E+00

          .00000E+00  .00000E+00  .26180E+00  .00000E+00  .65450E-01  .00000E+00  .00000E+00  .00000E+00


ASSEMBLING AND LINEAR SOLUTION (M= 0)
=============================
         INDEX FOR RESIDUAL COMPUTATION    (NRES)=    1
        ENERGY    (ENERG)=  .42653E+00

        ABSOLUTE VALUE OF MINIMUM PIVOT   =  .97469E+00 EQUATION:    5
                     ALGEBRAIC VALUE=  .97469E+00 EQUATION:    5
                     DETERMINANT       =  .14103E+09 * 10 **     0


MAX. RESIDUAL VALUE=  .26262E-15 EQUATION    6
```

SOLUTION

NODES	X	Y	Z	DEGREES OF FREEDOM (* = PRESCRIBED)
1	.00000E+00	.50000E+00	.00000E+00	.57520E+00
2	.00000E+00	.75000E+00	.00000E+00	.44681E+00
3	.00000E+00	.10000E+01	.00000E+00	.37137E+00
4	.00000E+00	.12500E+01	.00000E+00	.33326E+00
5	.00000E+00	.15000E+01	.00000E+00	.32317E+00
6	.19135E+00	.46195E+00	.00000E+00	.56684E+00
7	.47068E+00	.98097E+00	.00000E+00	.31756E+00
8	.75000E+00	.15000E+01	.00000E+00	.21866E+00
9	.35350E+00	.35350E+00	.00000E+00	.53887E+00
10	.64013E+00	.64013E+00	.00000E+00	.33300E+00
11	.92675E+00	.92675E+00	.00000E+00	.19818E+00
12	.12134E+01	.12134E+01	.00000E+00	.93103E-01
13	.15000E+01	.15000E+01	.00000E+00	.00000E+00 *
14	.46195E+00	.19135E+00	.00000E+00	.52269E+00
15	.98097E+00	.47068E+00	.00000E+00	.21421E+00
16	.15000E+01	.75000E+00	.00000E+00	.00000E+00 *
17	.50000E+00	.00000E+00	.00000E+00	.50587E+00
18	.75000E+00	.00000E+00	.00000E+00	.35749E+00
19	.10000E+01	.00000E+00	.00000E+00	.22597E+00
20	.12500E+01	.00000E+00	.00000E+00	.10981E+00
21	.15000E+01	.00000E+00	.00000E+00	.00000E+00 *

GRADIENTS IN ELEMENT : 1

```
P.G. : 1 COORDINATES :  .52632E-01  .55406E+00
         GRADIENTS   : -.11372E+00 -.81069E+00
P.G. : 2 COORDINATES :  .76523E-01  .74818E+00
         GRADIENTS   : -.12315E+00 -.57158E+00
P.G. : 3 COORDINATES :  .10041E+00  .94230E+00
         GRADIENTS   : -.11487E+00 -.33465E+00
P.G. : 4 COORDINATES :  .22283E+00  .52044E+00
         GRADIENTS   : -.41886E+00 -.70605E+00
P.G. : 5 COORDINATES :  .33101E+00  .72146E+00
         GRADIENTS   : -.33351E+00 -.49289E+00
P.G. : 6 COORDINATES :  .43919E+00  .92248E+00
         GRADIENTS   : -.28656E+00 -.25907E+00
P.G. : 7 COORDINATES :  .37650E+00  .44697E+00
         GRADIENTS   : -.65190E+00 -.47047E+00
P.G. : 8 COORDINATES :  .57236E+00  .66307E+00
         GRADIENTS   : -.51763E+00 -.32390E+00
P.G. : 9 COORDINATES :  .76823E+00  .87916E+00
         GRADIENTS   : -.44680E+00 -.11983E+00
```

GRADIENTS IN ELEMENT : 2

```
P.G. : 1 COORDINATES :  .44697E+00  .37650E+00
         GRADIENTS   : -.71704E+00 -.40960E+00
P.G. : 2 COORDINATES :  .66307E+00  .57236E+00
         GRADIENTS   : -.60362E+00 -.25620E+00
P.G. : 3 COORDINATES :  .87916E+00  .76823E+00
         GRADIENTS   : -.45595E+00 -.14059E+00
P.G. : 4 COORDINATES :  .52044E+00  .22283E+00
         GRADIENTS   : -.88575E+00 -.24160E+00
P.G. : 5 COORDINATES :  .72146E+00  .33101E+00
         GRADIENTS   : -.73534E+00 -.17979E+00
P.G. : 6 COORDINATES :  .92248E+00  .43919E+00
         GRADIENTS   : -.59060E+00 -.10744E+00
P.G. : 7 COORDINATES :  .55406E+00  .52632E-01
         GRADIENTS   : -.89617E+00 -.17241E-01
P.G. : 8 COORDINATES :  .74818E+00  .76523E-01
         GRADIENTS   : -.78959E+00 -.75879E-01
P.G. : 9 COORDINATES :  .94230E+00  .10041E+00
         GRADIENTS   : -.69255E+00 -.56986E-01
```

GRADIENTS IN ELEMENT : 3

```
P.G. : 1 COORDINATES :  .11432E+00  .10553E+01
         GRADIENTS   : -.11051E+00 -.25454E+00
P.G. : 2 COORDINATES :  .13821E+00  .12494E+01
         GRADIENTS   : -.11648E+00 -.14037E+00
P.G. : 3 COORDINATES :  .16210E+00  .14435E+01
         GRADIENTS   : -.13120E+00 -.25123E-01
P.G. : 4 COORDINATES :  .50216E+00  .10395E+01
         GRADIENTS   : -.27711E+00 -.20122E+00
P.G. : 5 COORDINATES :  .61034E+00  .12405E+01
         GRADIENTS   : -.28059E+00 -.11578E+00
P.G. : 6 COORDINATES :  .71852E+00  .14415E+01
         GRADIENTS   : -.29573E+00 -.24060E-01
P.G. : 7 COORDINATES :  .88222E+00  .10049E+01
         GRADIENTS   : -.43817E+00 -.86922E-01
P.G. : 8 COORDINATES :  .10781E+01  .12210E+01
         GRADIENTS   : -.44335E+00 -.28634E-01
P.G. : 9 COORDINATES :  .12740E+01  .14371E+01
         GRADIENTS   : -.46020E+00  .40218E-01
```

GRADIENTS IN ELEMENT : 4

```
P.G. : 1 COORDINATES :  .10049E+01  .88222E+00
         GRADIENTS   : -.46122E+00 -.99442E-01
P.G. : 2 COORDINATES :  .12210E+01  .10781E+01
         GRADIENTS   : -.46380E+00 -.47956E-01
P.G. : 3 COORDINATES :  .14371E+01  .12740E+01
         GRADIENTS   : -.45461E+00 -.94565E-02
P.G. : 4 COORDINATES :  .10395E+01  .50216E+00
         GRADIENTS   : -.57615E+00 -.74187E-01
P.G. : 5 COORDINATES :  .12405E+01  .61034E+00
         GRADIENTS   : -.55835E+00 -.36136E-01
P.G. : 6 COORDINATES :  .14415E+01  .71852E+00
         GRADIENTS   : -.53562E+00 -.72341E-02
P.G. : 7 COORDINATES :  .10553E+01  .11432E+00
         GRADIENTS   : -.65031E+00 -.38270E-01
P.G. : 8 COORDINATES :  .12494E+01  .13821E+00
         GRADIENTS   : -.62235E+00 -.20090E-01
P.G. : 9 COORDINATES :  .14435E+01  .16210E+00
         GRADIENTS   : -.59409E+00 -.42941E-02
```

EQUILIBRIUM RESIDUALS AND REACTIONS

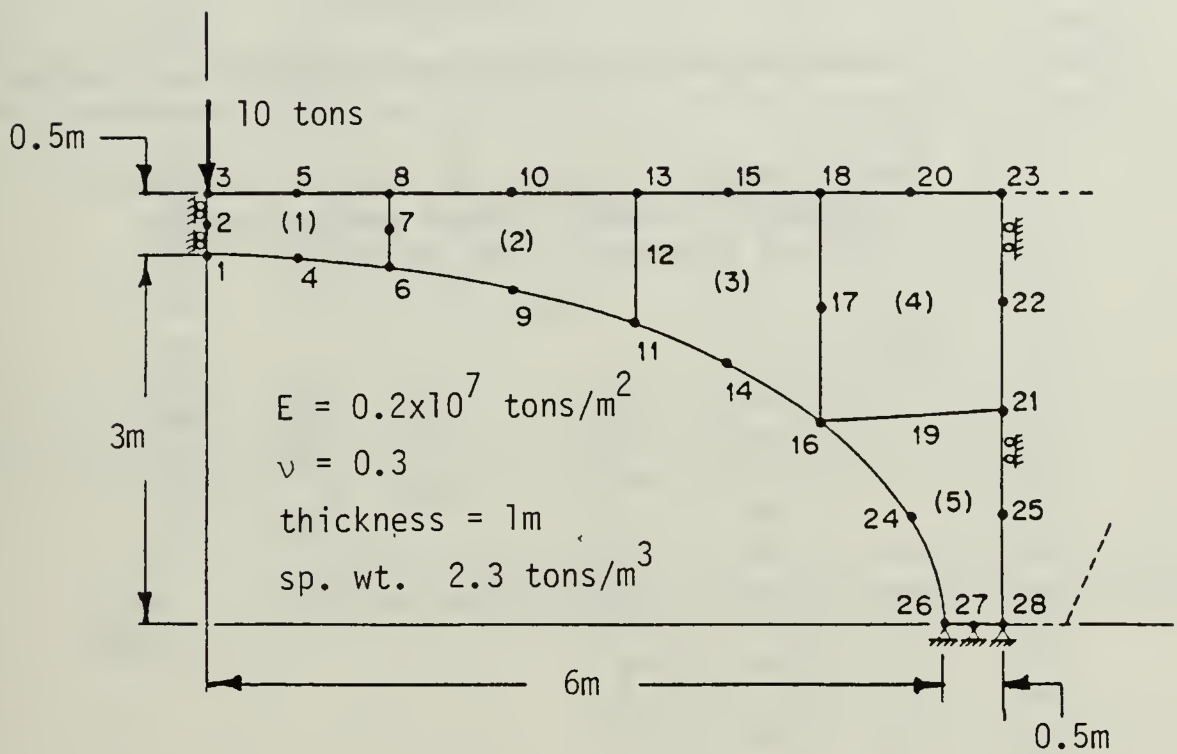
NODES	X	Y	Z	DEGREES OF FREEDOM (* = PRESCRIBED)
1	.00000E+00	.50000E+00	.00000E+00	.00000E+00
2	.00000E+00	.75000E+00	.00000E+00	-.10399E-15
3	.00000E+00	.10000E+01	.00000E+00	.27756E-16
4	.00000E+00	.12500E+01	.00000E+00	-.77743E-16
5	.00000E+00	.15000E+01	.00000E+00	.13109E-15
6	.19135E+00	.46195E+00	.00000E+00	-.27756E-15
7	.47068E+00	.98097E+00	.00000E+00	-.11102E-15
8	.75000E+00	.15000E+01	.00000E+00	-.63315E-16
9	.35350E+00	.35350E+00	.00000E+00	.37470E-15
10	.64013E+00	.64013E+00	.00000E+00	-.40246E-15
11	.92675E+00	.92675E+00	.00000E+00	.30531E-15
12	.12134E+01	.12134E+01	.00000E+00	.19429E-15
13	.15000E+01	.15000E+01	.00000E+00	-.10288E+00 *
14	.46195E+00	.19135E+00	.00000E+00	.11102E-15

15	.98097E+00	.47068E+00	.00000E+00	.11102E-15
16	.15000E+01	.75000E+00	.00000E+00	-.52531E+00 *
17	.50000E+00	.00000E+00	.00000E+00	-.13878E-16
18	.75000E+00	.00000E+00	.00000E+00	-.25562E-15
19	.10000E+01	.00000E+00	.00000E+00	.69389E-16
20	.12500E+01	.00000E+00	.00000E+00	.10714E-15
21	.15000E+01	.00000E+00	.00000E+00	-.15722E+00 *

END OF PROBLEM,   314 UTILIZED REAL WORDS OVER   20000

Concrete elliptical arch in plane stress, for comparison with the results of Dhatt and Touzot.

EIGEN VALUE PROBLEM



The loads consist of the distributed dead weight, and the concentrated force of 10 tons at node 3.

F.E.M.3.
G.TOUZOT, G.DHATT
MODIFIED BY

REHE E. RUESCH

IMAGE OF DATA CARDS

```
                              C O L U M N   N U M B E R
 CARD               1         2         3         4         5         6         7         8
NUMBER     12345678901234567890123456789012345678901234567890123456789012345678901234567890
--------   --------------------------------------------------------------------------------
    1      COMT
    2        ELASTIC ANALYSIS OF AN ELLIPTIC HALF BRIDGE ARCH IN PLANE STRESS
    3        SAMPLE PROBLEM TO COMPARE THE RESULTS OF MEF ON THE IBM PC
    4          WITH THOSE OF THE AUTHORS, DHATT AND TOUZOT
    5
    6      COOR
    7         28    2    2
    8          3      0.00      3.50             23      6.50      3.50              5
    9          5      0.75      3.50             20      5.75      3.50              5
   10          2      0.00      3.25
   11          7      1.50      3.20
   12         12      3.50      2.97
   13         17      5.00      2.58
   14         19      5.75      1.70
   15          1      0.00      3.00
   16          4      0.75      2.98
   17          6      1.50      2.90
   18          9      2.50      2.73
   19         11      3.50      2.44
   20         14      4.25      2.12
   21         16      5.00      1.66
   22         24      5.75      0.86
   23         26      6.00      0.00
   24         27      6.25      0.00
   25         28      6.50      0.00
   26         25      6.50      0.87
   27         21      6.50      1.75
--------   --------------------------------------------------------------------------------
 CARD      12345678901234567890123456789012345678901234567890123456789012345678901234567890
NUMBER              1         2         3         4         5         6         7         8
                              C O L U M N   N U M B E R
```

```
                              C O L U M N   N U M B E R
  CARD             1         2         3         4         5         6         7         8
 NUMBER   12345678901234567890123456789012345678901234567890123456789012345678901234567890
 ------   --------------------------------------------------------------------------------
     28      22      6.50      2.62
     29      -1
     30   COND
     31   11
     32      26   27   28
     33   10
     34       1    2    3   25   21   22   23
     35   0000000000
     36   PREL
     37       1    4
     38       1     2.0E6 0.3       0.0       2.3
     39      -1
     40   ELEM
     41       5    8    2    1    0
     42       1    4    5    0    1    0    1    4    6    7    8    5    3    2
     43       5    1    0    0    1    0   26   27   28   25   21   19   16   24
     44      -1
     45   SOLC
     46       1      0.00    -10.00
     47       3
     48      -1
     49   SOLR
     50   VALP
     51       3   20 0.001      0.0         5    0   12    1.D-12
     52   STOP
 ------   --------------------------------------------------------------------------------
  CARD    12345678901234567890123456789012345678901234567890123456789012345678901234567890
 NUMBER            1         2         3         4         5         6         7         8
                              C O L U M N   N U M B E R
```

E N D   O F   D A T A

COMMENTS
==========

ELASTIC ANALYSIS OF AN ELLIPTIC HALF BRIDGE ARCH IN PLANE STRESS
SAMPLE PROBLEM TO COMPARE THE RESULTS OF MEF ON THE IBM PC
WITH THOSE OF THE AUTHORS, DHATT AND TOUZOT

INPUT OF NODES (M= 0)

================

MAX. NUMBER OF NODES (NNT)= 28
MAX. NUMBER OF D.O.F. PER NODE (NDLN)= 2
DIMENSIONS OF THE PROBLEM (NDIM)= 2
COORDINATE SCALE FACTORS (FAC)= .10000E+01 .10000E+01 .10000E+01
WORKSPACE IN REAL WORDS (NVA)= 20000

INPUT OF BOUNDARY CONDITIONS (M= 0)
=============================

BOUNDARY CONDITIONS CARDS

```
)))))1100000000  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00
)))))             26   27   28    0    0    0    0    0    0    0    0    0    0    0    0    0
)))))1000000000  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00
)))))              1    2    3   25   21   22   23    0    0    0    0    0    0    0    0    0
)))))0000000000  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00
```

TOTAL NUMBER OF NODES (NNT)= 28
TOTAL NUMBER OF D.O.F. (NDLT)= 56
NUMBER OF EQUATIONS TO BE SOLVED (NEQ)= 43
NUMBER OF PRESCRIBED NON ZERO D.O.F. (NCLNZ)= 0
NUMBER OF PRESCRIBED ZERO D.O.F. (NCLZ)= 13
TOTAL NUMBER OF PRESCRIBED D.O.F. (NCLT)= 13

NODAL COORDINATES ARRAY

NO	D.L.	X	Y	Z	EQUATION NUMBER		(NEQ)
1	2	.00000E+00	.30000E+01	.00000E+00	-7	1	
2	2	.00000E+00	.32500E+01	.00000E+00	-8	2	
3	2	.00000E+00	.35000E+01	.00000E+00	-9	3	
4	2	.75000E+00	.29800E+01	.00000E+00	4	5	
5	2	.75000E+00	.35000E+01	.00000E+00	6	7	
6	2	.15000E+01	.29000E+01	.00000E+00	8	9	
7	2	.15000E+01	.32000E+01	.00000E+00	10	11	
8	2	.16250E+01	.35000E+01	.00000E+00	12	13	
9	2	.25000E+01	.27300E+01	.00000E+00	14	15	
10	2	.24167E+01	.35000E+01	.00000E+00	16	17	
11	2	.35000E+01	.24400E+01	.00000E+00	18	19	
12	2	.35000E+01	.29700E+01	.00000E+00	20	21	
13	2	.32500E+01	.35000E+01	.00000E+00	22	23	
14	2	.42500E+01	.21200E+01	.00000E+00	24	25	
15	2	.40833E+01	.35000E+01	.00000E+00	26	27	
16	2	.50000E+01	.16600E+01	.00000E+00	28	29	

```
 17   2  .50000E+01  .25800E+01  .00000E+00          30   31
 18   2  .48750E+01  .35000E+01  .00000E+00          32   33
 19   2  .57500E+01  .17000E+01  .00000E+00          34   35
 20   2  .57500E+01  .35000E+01  .00000E+00          36   37
 21   2  .65000E+01  .17500E+01  .00000E+00         -11   38
 22   2  .65000E+01  .26200E+01  .00000E+00         -12   39
 23   2  .65000E+01  .35000E+01  .00000E+00         -13   40
 24   2  .57500E+01  .86000E+00  .00000E+00          41   42
 25   2  .65000E+01  .87000E+00  .00000E+00         -10   43
 26   2  .60000E+01  .00000E+00  .00000E+00          -1   -2
 27   2  .62500E+01  .00000E+00  .00000E+00          -3   -4
 28   2  .65000E+01  .00000E+00  .00000E+00          -5   -6


INPUT OF ELEMENT PROPERTIES (M= 0)
==================================
          NUMBER OF GROUPS OF PROPERTIES   (NGPE)=    1
          NUMBER OF PROPERTIES PER GROUP   (NPRE)=    4


CARDS OF ELEMENT PROPERTIES

)))))    1  .20000E+07  .30000E+00  .00000E+00  .23000E+01
)))))   -1  .00000E+00  .00000E+00  .00000E+00  .00000E+00


INPUT OF ELEMENTS (M= 0)
========================
          MAX. NUMBER OF ELEMENTS            (NELT)=    5
          MAX. NUMBER OF NODES PER ELEMENT   (NNEL)=    8
          DEFAULT ELEMENT TYPE               (NTPE)=    2
          NUMBER OF GROUPS OF ELEMENTS       (NGRE)=    1
          INDEX FOR NON SYMMETRIC PROBLEM    (NSYM)=    0
          INDEX FOR IDENTICAL ELEMENTS     (NIDENT)=    0

      ELEMENT:    1 TYPE: 2 N.P.: 8 D.O.F.: 16 N. PROP:  0 EL. PROP:  4 GROUP:  0
          CONNECTIVITY (NE)    1    4    6    7    8    5    3    2
      ELEMENT:    2 TYPE: 2 N.P.: 8 D.O.F.: 16 N. PROP:  0 EL. PROP:  4 GROUP:  0
          CONNECTIVITY (NE)    6    9   11   12   13   10    8    7
      ELEMENT:    3 TYPE: 2 N.P.: 8 D.O.F.: 16 N. PROP:  0 EL. PROP:  4 GROUP:  0
          CONNECTIVITY (NE)   11   14   16   17   18   15   13   12
      ELEMENT:    4 TYPE: 2 N.P.: 8 D.O.F.: 16 N. PROP:  0 EL. PROP:  4 GROUP:  0
          CONNECTIVITY (NE)   16   19   21   22   23   20   18   17
      ELEMENT:    5 TYPE: 2 N.P.: 8 D.O.F.: 16 N. PROP:  0 EL. PROP:  4 GROUP:  0
          CONNECTIVITY (NE)   26   27   28   25   21   19   16   24

          MEAN BAND HEIGHT=     9.1 MAXIMUM=    15
          LENGTH OF A TRIANGLE IN KG         (NKG)=      393
          NUMBER OF INTEGRATION POINTS       (NPG)=       45
```

INPUT OF CONCENTRADED LOADS (M= 0)
===================================

CARDS OF NODAL LOADS

```
)))))    1  .00000E+00 -.10000E+02
)))))    3    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
)))))   -1  .00000E+00  .00000E+00
```

ASSEMBLING OF DISTRIBUTED LOADS (M= 0)
======================================

SUBSPACE ITERATION (M= 0)
=========================

```
              NUMBER OF DESIRED EIGENVALUES            (NVALP)=           3
              MAX. NUMBER OF ITERATIONS PERMITTED      (NITER)=          20
              INDEX FOR DIAGONAL MATRIX               (NMDIAG)=           0
              CONVERGENCE TOLERANCE ON EIGENVALUES     (EPSLB)=  .10000E-02
              SHIFT                                    (SHIFT)=  .00000E+00
              SUBSPACE DIMENSION                         (NSS)=           5
              MAX. NUMBER OF ITERATION IN JACOBI        (NSWM)=          12
              CONVERGENCE TOLERANCE IN JACOBI         (TOLJAC)=  .10000E-11

ITERATION    1 MAX. ERROR=   .5E+06 EXACT EIGENVALUES:   0
ITERATION    2 MAX. ERROR=   .4E+00 EXACT EIGENVALUES:   0
ITERATION    3 MAX. ERROR=   .1E-01 EXACT EIGENVALUES:   3

. . . . CONVERGENCE IN   4 ITERATIONS


EIGENVALUE NO.     1 =  .56152E+04

EIGENVECTOR:

NODES    X            Y            Z            DEGREES OF FREEDOM (* = PRESCRIBED)

    1  .00000E+00  .30000E+01  .00000E+00       .00000E+00 *    .66375E+00
    2  .00000E+00  .32500E+01  .00000E+00       .00000E+00 *    .66172E+00
    3  .00000E+00  .35000E+01  .00000E+00       .00000E+00 *    .65867E+00
    4  .75000E+00  .29800E+01  .00000E+00      -.19930E-01      .60786E+00
    5  .75000E+00  .35000E+01  .00000E+00       .45051E-01      .60490E+00
    6  .15000E+01  .29000E+01  .00000E+00      -.40117E-01      .48147E+00
    7  .15000E+01  .32000E+01  .00000E+00       .18612E-01      .47890E+00
    8  .16250E+01  .35000E+01  .00000E+00       .78526E-01      .45397E+00
    9  .25000E+01  .27300E+01  .00000E+00      -.43577E-01      .28471E+00
```

10	.24167E+01	.35000E+01	.00000E+00	.80294E-01	.29658E+00
11	.35000E+01	.24400E+01	.00000E+00	-.43399E-01	.14162E+00
12	.35000E+01	.29700E+01	.00000E+00	.72541E-02	.13794E+00
13	.32500E+01	.35000E+01	.00000E+00	.68736E-01	.17225E+00
14	.42500E+01	.21200E+01	.00000E+00	-.32304E-01	.72376E-01
15	.40833E+01	.35000E+01	.00000E+00	.48682E-01	.85795E-01
16	.50000E+01	.16600E+01	.00000E+00	-.17548E-01	.30240E-01
17	.50000E+01	.25800E+01	.00000E+00	-.50562E-02	.33255E-01
18	.48750E+01	.35000E+01	.00000E+00	.29860E-01	.39282E-01
19	.57500E+01	.17000E+01	.00000E+00	-.78157E-02	.17301E-01
20	.57500E+01	.35000E+01	.00000E+00	.14539E-01	.16647E-01
21	.65000E+01	.17500E+01	.00000E+00	.00000E+00 *	.11601E-01
22	.65000E+01	.26200E+01	.00000E+00	.00000E+00 *	.95616E-02
23	.65000E+01	.35000E+01	.00000E+00	.00000E+00 *	.11585E-01
24	.57500E+01	.86000E+00	.00000E+00	-.19224E-02	.13207E-01
25	.65000E+01	.87000E+00	.00000E+00	.00000E+00 *	.98898E-02
26	.60000E+01	.00000E+00	.00000E+00	.00000E+00 *	.00000E+00 *
27	.62500E+01	.00000E+00	.00000E+00	.00000E+00 *	.00000E+00 *
28	.65000E+01	.00000E+00	.00000E+00	.00000E+00 *	.00000E+00 *

EIGENVALUE NO. 2 = .37888E+05

EIGENVECTOR:

NODES	X	Y	Z	DEGREES OF FREEDOM (* = PRESCRIBED)	
1	.00000E+00	.30000E+01	.00000E+00	.00000E+00 *	-.43557E+00
2	.00000E+00	.32500E+01	.00000E+00	.00000E+00 *	-.43947E+00
3	.00000E+00	.35000E+01	.00000E+00	.00000E+00 *	-.43308E+00
4	.75000E+00	.29800E+01	.00000E+00	.75478E-01	-.30806E+00
5	.75000E+00	.35000E+01	.00000E+00	-.66589E-01	-.30728E+00
6	.15000E+01	.29000E+01	.00000E+00	.12568E+00	-.50310E-01
7	.15000E+01	.32000E+01	.00000E+00	.19733E-01	-.43091E-01
8	.16250E+01	.35000E+01	.00000E+00	-.83712E-01	-.50229E-02
9	.25000E+01	.27300E+01	.00000E+00	.10301E+00	.25248E+00
10	.24167E+01	.35000E+01	.00000E+00	-.30655E-01	.23703E+00
11	.35000E+01	.24400E+01	.00000E+00	.43413E-01	.31596E+00
12	.35000E+01	.29700E+01	.00000E+00	.53209E-01	.32955E+00
13	.32500E+01	.35000E+01	.00000E+00	.32536E-01	.31620E+00
14	.42500E+01	.21200E+01	.00000E+00	.34673E-02	.27207E+00
15	.40833E+01	.35000E+01	.00000E+00	.74869E-01	.29474E+00
16	.50000E+01	.16600E+01	.00000E+00	-.12682E-01	.20050E+00
17	.50000E+01	.25800E+01	.00000E+00	.22806E-01	.21003E+00
18	.48750E+01	.35000E+01	.00000E+00	.77638E-01	.23648E+00
19	.57500E+01	.17000E+01	.00000E+00	-.23282E-02	.14099E+00
20	.57500E+01	.35000E+01	.00000E+00	.40232E-01	.17473E+00
21	.65000E+01	.17500E+01	.00000E+00	.00000E+00 *	.11596E+00
22	.65000E+01	.26200E+01	.00000E+00	.00000E+00 *	.14199E+00
23	.65000E+01	.35000E+01	.00000E+00	.00000E+00 *	.15438E+00
24	.57500E+01	.86000E+00	.00000E+00	.15066E-02	.10386E+00
25	.65000E+01	.87000E+00	.00000E+00	.00000E+00 *	.71782E-01

26	.60000E+01	.00000E+00	.00000E+00	.00000E+00 *	.00000E+00 *
27	.62500E+01	.00000E+00	.00000E+00	.00000E+00 *	.00000E+00 *
28	.65000E+01	.00000E+00	.00000E+00	.00000E+00 *	.00000E+00 *

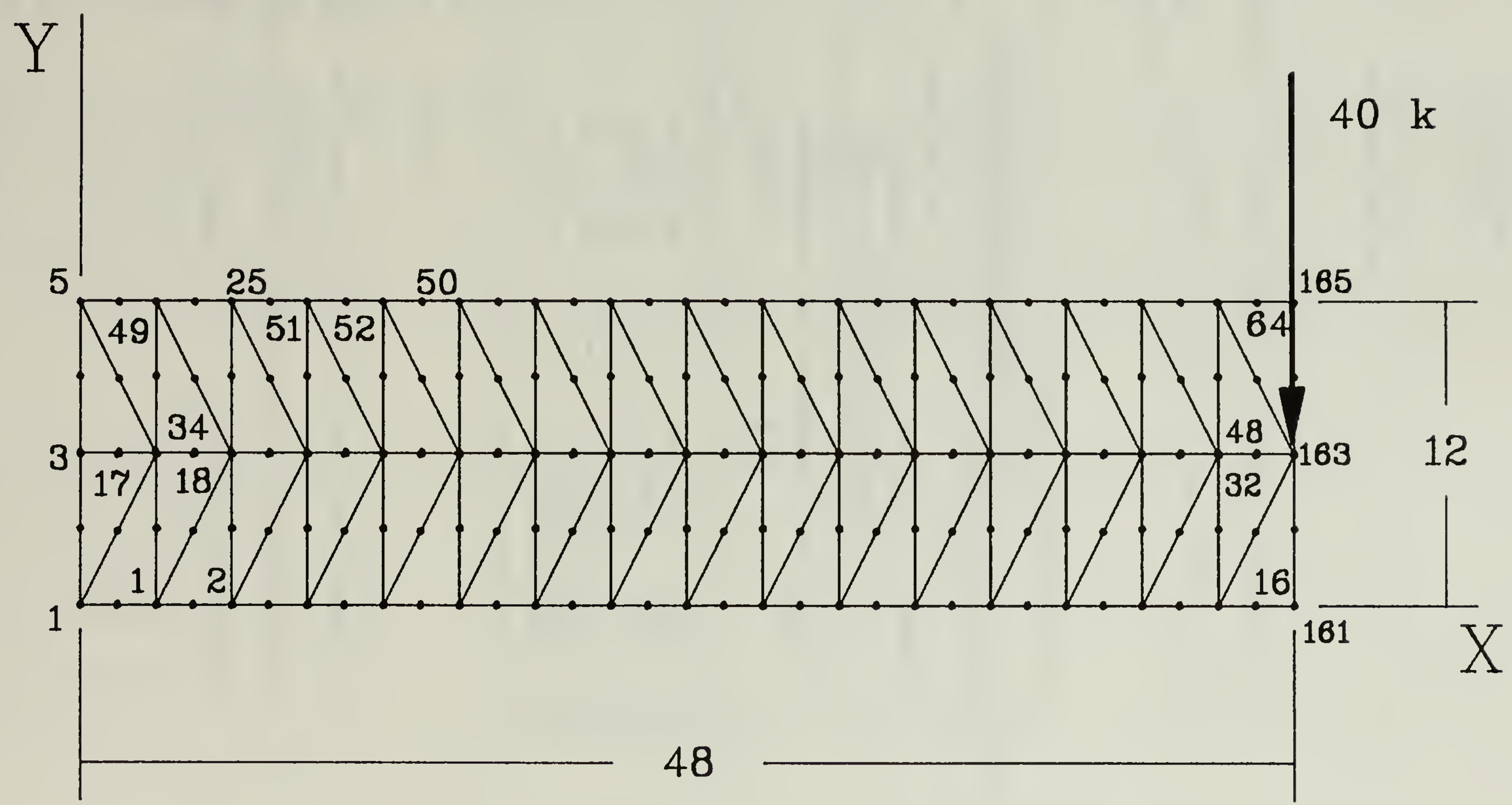
EIGENVALUE NO. 3 = .10014E+06

EIGENVECTOR:

NODES	X	Y	Z	DEGREES OF FREEDOM (* = PRESCRIBED)	
1	.00000E+00	.30000E+01	.00000E+00	.00000E+00 *	-.41128E+00
2	.00000E+00	.32500E+01	.00000E+00	.00000E+00 *	-.42594E+00
3	.00000E+00	.35000E+01	.00000E+00	.00000E+00 *	-.41626E+00
4	.75000E+00	.29800E+01	.00000E+00	.12574E+00	-.22292E+00
5	.75000E+00	.35000E+01	.00000E+00	-.69262E-01	-.22541E+00
6	.15000E+01	.29000E+01	.00000E+00	.17483E+00	.10569E+00
7	.15000E+01	.32000E+01	.00000E+00	.68934E-01	.11778E+00
8	.16250E+01	.35000E+01	.00000E+00	-.33533E-01	.14924E+00
9	.25000E+01	.27300E+01	.00000E+00	.89392E-01	.32437E+00
10	.24167E+01	.35000E+01	.00000E+00	.86100E-01	.32239E+00
11	.35000E+01	.24400E+01	.00000E+00	-.47209E-01	.12451E+00
12	.35000E+01	.29700E+01	.00000E+00	.81544E-01	.14886E+00
13	.32500E+01	.35000E+01	.00000E+00	.18077E+00	.20020E+00
14	.42500E+01	.21200E+01	.00000E+00	-.87810E-01	-.78360E-01
15	.40833E+01	.35000E+01	.00000E+00	.18541E+00	-.14116E-01
16	.50000E+01	.16600E+01	.00000E+00	-.75138E-01	-.21317E+00
17	.50000E+01	.25800E+01	.00000E+00	-.10474E-01	-.21304E+00
18	.48750E+01	.35000E+01	.00000E+00	.13544E+00	-.17181E+00
19	.57500E+01	.17000E+01	.00000E+00	-.41600E-01	-.21638E+00
20	.57500E+01	.35000E+01	.00000E+00	.62931E-01	-.27285E+00
21	.65000E+01	.17500E+01	.00000E+00	.00000E+00 *	-.22656E+00
22	.65000E+01	.26200E+01	.00000E+00	.00000E+00 *	-.28436E+00
23	.65000E+01	.35000E+01	.00000E+00	.00000E+00 *	-.29085E+00
24	.57500E+01	.86000E+00	.00000E+00	-.27562E-01	-.16051E+00
25	.65000E+01	.87000E+00	.00000E+00	.00000E+00 *	-.11610E+00
26	.60000E+01	.00000E+00	.00000E+00	.00000E+00 *	.00000E+00 *
27	.62500E+01	.00000E+00	.00000E+00	.00000E+00 *	.00000E+00 *
28	.65000E+01	.00000E+00	.00000E+00	.00000E+00 *	.00000E+00 *

END OF PROBLEM, 1676 UTILIZED REAL WORDS OVER 20000

E = 30,000 ksi
ν = 0.25
thickness = 1
All dimensions in inches



cantilevered beam for comparison with the results of Felippa

The following table of results for the cantilevered beam was obtained by Carlos A. Felippa.

DEFLECTION AND NORMAL STRESS			
Element	Mesh	Tip Deflection $\delta = v_C$	Stress σ_x at X = 9", Y = 6"
CST	A-1 A-2	0.30556 0.34188	51.225 57.342
LST	B-1 B-2	0.35506 0.35569	59.145 60.024
Beam Theory (upper bound for v_C)		0.35583	60.000

For comparison, the tip defelections from MEF are the ones for nodes 161 thru 165. The stresses are the stresses listed for Gauss Point 3 of elements 51 and 52.

F.E.M.3.
G.TOUZOT, G.DHATT
MODIFIED BY

REHE E. RUESCH

IMAGE OF DATA CARDS

```
                                      C O L U M N   N U M B E R
 CARD               1         2         3         4         5         6         7         8
 NUMBER    12345678901234567890123456789012345678901234567890123456789012345678901234567890
 ------    --------------------------------------------------------------------------------
     1     COMT
     2         CANTILEAVERED BEAM 12 X 48 X 1 (INCHES)
     3         6 NODED TRIANGUALAR ELEMENTS (ELEMO3)
     4         64 ELEMENTS
     5         FOR COMPARISON WITH THE RESULTS OF CARLOS A. FELIPPA
     6
     7     COOR
     8       165    2    2
     9         1        0.0        0.0       0.0  161      48.0       0.0       0.0   5
    10         2        0.0        3.0       0.0  162      48.0       3.0       0.0   5
    11         3        0.0        6.0       0.0  163      48.0       6.0       0.0   5
    12         4        0.0        9.0       0.0  164      48.0       9.0       0.0   5
    13         5        0.0       12.0       0.0  165      48.0      12.0       0.0   5
    14     0
    15     COND
    16     11
    17     1,2,3,4,5,0
    18     0
    19     PREL
    20         1    3
    21     1,30.0E03,0.25,0.0,
    22     0
    23     ELEM
    24        64    6    3
    25         1   16   10    3    1    1    1    6   11   12   13    7    0
    26        17   16   10    3    1    1    1    7   13    8    3    2    0
    27        33   16   10    3    1    1    3    8   13    9    5    4    0
    28        49   16   10    3    1    1    5    9   13   14   15   10    0
    29     0
    30     SOLC
    31     1,0.0,-40.0,0.0
```

```
     32       163.
     33       0
     34       LINM
     35          1
     36       STOP
  -------     ------------------------------------------------------------------------------
   CARD       12345678901234567890123456789012345678901234567890123456789012345678901234567890
  NUMBER               1         2         3         4         5         6         7         8
                                       C O L U M N   N U M B E R
```

E N D   O F   D A T A

COMMENTS
==========

CANTILEAVERED BEAM 12 X 48 X 1 (INCHES)
6 NODED TRIANGUALAR ELEMENTS (ELEM03)
64 ELEMENTS
FOR COMPARISON WITH THE RESULTS OF CARLOS A. FELIPPA

INPUT OF NODES (M= 0)
=================

```
          MAX. NUMBER OF NODES                (NNT)=  165
          MAX. NUMBER OF D.O.F. PER NODE     (NDLN)=    2
          DIMENSIONS OF THE PROBLEM          (NDIM)=    2
          COORDINATE SCALE FACTORS            (FAC)=  .10000E+01  .10000E+01  .10000E+01
          WORKSPACE IN REAL WORDS             (NVA)=     20000
```

INPUT OF BOUNDARY CONDITIONS (M= 0)
===================================

BOUNDARY CONDITIONS CARDS

```
>>>>>1100000000  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00
>>>>>              1    2    3    4    5    0    0    0    0    0    0    0    0    0    0    0
>>>>>0000000000  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00  .00000E+00
```

```
          TOTAL NUMBER OF NODES                        (NNT)=  165
          TOTAL NUMBER OF D.O.F.                      (NDLT)=  330
          NUMBER OF EQUATIONS TO BE SOLVED             (NEQ)=  320
          NUMBER OF PRESCRIBED NON ZERO D.O.F.       (NCLNZ)=    0
          MUMBER OF PRESCRIBED ZERO D.O.F.            (NCLZ)=   10
```

TOTAL NUMBER OF PRESCRIBED D.O.F. (NCLT)= 10

NODAL COORDINATES ARRAY

NO	D.L.	X	Y	Z	EQUATION NUMBER	(NEQ)
1	2	.00000E+00	.00000E+00	.00000E+00	-1	-2
2	2	.00000E+00	.30000E+01	.00000E+00	-3	-4
3	2	.00000E+00	.60000E+01	.00000E+00	-5	-6
4	2	.00000E+00	.90000E+01	.00000E+00	-7	-8
5	2	.00000E+00	.12000E+02	.00000E+00	-9	-10
6	2	.15000E+01	.00000E+00	.00000E+00	1	2
7	2	.15000E+01	.30000E+01	.00000E+00	3	4
8	2	.15000E+01	.60000E+01	.00000E+00	5	6
9	2	.15000E+01	.90000E+01	.00000E+00	7	8
10	2	.15000E+01	.12000E+02	.00000E+00	9	10
11	2	.30000E+01	.00000E+00	.00000E+00	11	12
12	2	.30000E+01	.30000E+01	.00000E+00	13	14
13	2	.30000E+01	.60000E+01	.00000E+00	15	16
14	2	.30000E+01	.90000E+01	.00000E+00	17	18
15	2	.30000E+01	.12000E+02	.00000E+00	19	20
16	2	.45000E+01	.00000E+00	.00000E+00	21	22
17	2	.45000E+01	.30000E+01	.00000E+00	23	24
18	2	.45000E+01	.60000E+01	.00000E+00	25	26
19	2	.45000E+01	.90000E+01	.00000E+00	27	28
20	2	.45000E+01	.12000E+02	.00000E+00	29	30
21	2	.60000E+01	.00000E+00	.00000E+00	31	32
22	2	.60000E+01	.30000E+01	.00000E+00	33	34
23	2	.60000E+01	.60000E+01	.00000E+00	35	36
24	2	.60000E+01	.90000E+01	.00000E+00	37	38
25	2	.60000E+01	.12000E+02	.00000E+00	39	40
26	2	.75000E+01	.00000E+00	.00000E+00	41	42
27	2	.75000E+01	.30000E+01	.00000E+00	43	44
28	2	.75000E+01	.60000E+01	.00000E+00	45	46
29	2	.75000E+01	.90000E+01	.00000E+00	47	48
30	2	.75000E+01	.12000E+02	.00000E+00	49	50
31	2	.90000E+01	.00000E+00	.00000E+00	51	52
32	2	.90000E+01	.30000E+01	.00000E+00	53	54
33	2	.90000E+01	.60000E+01	.00000E+00	55	56
34	2	.90000E+01	.90000E+01	.00000E+00	57	58
35	2	.90000E+01	.12000E+02	.00000E+00	59	60
36	2	.10500E+02	.00000E+00	.00000E+00	61	62
37	2	.10500E+02	.30000E+01	.00000E+00	63	64
38	2	.10500E+02	.60000E+01	.00000E+00	65	66
39	2	.10500E+02	.90000E+01	.00000E+00	67	68
40	2	.10500E+02	.12000E+02	.00000E+00	69	70
41	2	.12000E+02	.00000E+00	.00000E+00	71	72
42	2	.12000E+02	.30000E+01	.00000E+00	73	74
43	2	.12000E+02	.60000E+01	.00000E+00	75	76

44	2	.12000E+02	.90000E+01	.00000E+00	77	78
45	2	.12000E+02	.12000E+02	.00000E+00	79	80
46	2	.13500E+02	.00000E+00	.00000E+00	81	82
47	2	.13500E+02	.30000E+01	.00000E+00	83	84
48	2	.13500E+02	.60000E+01	.00000E+00	85	86
49	2	.13500E+02	.90000E+01	.00000E+00	87	88
50	2	.13500E+02	.12000E+02	.00000E+00	89	90
51	2	.15000E+02	.00000E+00	.00000E+00	91	92
52	2	.15000E+02	.30000E+01	.00000E+00	93	94
53	2	.15000E+02	.60000E+01	.00000E+00	95	96
54	2	.15000E+02	.90000E+01	.00000E+00	97	98
55	2	.15000E+02	.12000E+02	.00000E+00	99	100
56	2	.16500E+02	.00000E+00	.00000E+00	101	102
57	2	.16500E+02	.30000E+01	.00000E+00	103	104
58	2	.16500E+02	.60000E+01	.00000E+00	105	106
59	2	.16500E+02	.90000E+01	.00000E+00	107	108
60	2	.16500E+02	.12000E+02	.00000E+00	109	110
61	2	.18000E+02	.00000E+00	.00000E+00	111	112
62	2	.18000E+02	.30000E+01	.00000E+00	113	114
63	2	.18000E+02	.60000E+01	.00000E+00	115	116
64	2	.18000E+02	.90000E+01	.00000E+00	117	118
65	2	.18000E+02	.12000E+02	.00000E+00	119	120
66	2	.19500E+02	.00000E+00	.00000E+00	121	122
67	2	.19500E+02	.30000E+01	.00000E+00	123	124
68	2	.19500E+02	.60000E+01	.00000E+00	125	126
69	2	.19500E+02	.90000E+01	.00000E+00	127	128
70	2	.19500E+02	.12000E+02	.00000E+00	129	130
71	2	.21000E+02	.00000E+00	.00000E+00	131	132
72	2	.21000E+02	.30000E+01	.00000E+00	133	134
73	2	.21000E+02	.60000E+01	.00000E+00	135	136
74	2	.21000E+02	.90000E+01	.00000E+00	137	138
75	2	.21000E+02	.12000E+02	.00000E+00	139	140
76	2	.22500E+02	.00000E+00	.00000E+00	141	142
77	2	.22500E+02	.30000E+01	.00000E+00	143	144
78	2	.22500E+02	.60000E+01	.00000E+00	145	146
79	2	.22500E+02	.90000E+01	.00000E+00	147	148
80	2	.22500E+02	.12000E+02	.00000E+00	149	150
81	2	.24000E+02	.00000E+00	.00000E+00	151	152
82	2	.24000E+02	.30000E+01	.00000E+00	153	154
83	2	.24000E+02	.60000E+01	.00000E+00	155	156
84	2	.24000E+02	.90000E+01	.00000E+00	157	158
85	2	.24000E+02	.12000E+02	.00000E+00	159	160
86	2	.25500E+02	.00000E+00	.00000E+00	161	162
87	2	.25500E+02	.30000E+01	.00000E+00	163	164
88	2	.25500E+02	.60000E+01	.00000E+00	165	166
89	2	.25500E+02	.90000E+01	.00000E+00	167	168
90	2	.25500E+02	.12000E+02	.00000E+00	169	170
91	2	.27000E+02	.00000E+00	.00000E+00	171	172
92	2	.27000E+02	.30000E+01	.00000E+00	173	174
93	2	.27000E+02	.60000E+01	.00000E+00	175	176
94	2	.27000E+02	.90000E+01	.00000E+00	177	178

95	2	.27000E+02	.12000E+02	.00000E+00	179	180
96	2	.28500E+02	.00000E+00	.00000E+00	181	182
97	2	.28500E+02	.30000E+01	.00000E+00	183	184
98	2	.28500E+02	.60000E+01	.00000E+00	185	186
99	2	.28500E+02	.90000E+01	.00000E+00	187	188
100	2	.28500E+02	.12000E+02	.00000E+00	189	190
101	2	.30000E+02	.00000E+00	.00000E+00	191	192
102	2	.30000E+02	.30000E+01	.00000E+00	193	194
103	2	.30000E+02	.60000E+01	.00000E+00	195	196
104	2	.30000E+02	.90000E+01	.00000E+00	197	198
105	2	.30000E+02	.12000E+02	.00000E+00	199	200
106	2	.31500E+02	.00000E+00	.00000E+00	201	202
107	2	.31500E+02	.30000E+01	.00000E+00	203	204
108	2	.31500E+02	.60000E+01	.00000E+00	205	206
109	2	.31500E+02	.90000E+01	.00000E+00	207	208
110	2	.31500E+02	.12000E+02	.00000E+00	209	210
111	2	.33000E+02	.00000E+00	.00000E+00	211	212
112	2	.33000E+02	.30000E+01	.00000E+00	213	214
113	2	.33000E+02	.60000E+01	.00000E+00	215	216
114	2	.33000E+02	.90000E+01	.00000E+00	217	218
115	2	.33000E+02	.12000E+02	.00000E+00	219	220
116	2	.34500E+02	.00000E+00	.00000E+00	221	222
117	2	.34500E+02	.30000E+01	.00000E+00	223	224
118	2	.34500E+02	.60000E+01	.00000E+00	225	226
119	2	.34500E+02	.90000E+01	.00000E+00	227	228
120	2	.34500E+02	.12000E+02	.00000E+00	229	230
121	2	.36000E+02	.00000E+00	.00000E+00	231	232
122	2	.36000E+02	.30000E+01	.00000E+00	233	234
123	2	.36000E+02	.60000E+01	.00000E+00	235	236
124	2	.36000E+02	.90000E+01	.00000E+00	237	238
125	2	.36000E+02	.12000E+02	.00000E+00	239	240
126	2	.37500E+02	.00000E+00	.00000E+00	241	242
127	2	.37500E+02	.30000E+01	.00000E+00	243	244
128	2	.37500E+02	.60000E+01	.00000E+00	245	246
129	2	.37500E+02	.90000E+01	.00000E+00	247	248
130	2	.37500E+02	.12000E+02	.00000E+00	249	250
131	2	.39000E+02	.00000E+00	.00000E+00	251	252
132	2	.39000E+02	.30000E+01	.00000E+00	253	254
133	2	.39000E+02	.60000E+01	.00000E+00	255	256
134	2	.39000E+02	.90000E+01	.00000E+00	257	258
135	2	.39000E+02	.12000E+02	.00000E+00	259	260
136	2	.40500E+02	.00000E+00	.00000E+00	261	262
137	2	.40500E+02	.30000E+01	.00000E+00	263	264
138	2	.40500E+02	.60000E+01	.00000E+00	265	266
139	2	.40500E+02	.90000E+01	.00000E+00	267	268
140	2	.40500E+02	.12000E+02	.00000E+00	269	270
141	2	.42000E+02	.00000E+00	.00000E+00	271	272
142	2	.42000E+02	.30000E+01	.00000E+00	273	274
143	2	.42000E+02	.60000E+01	.00000E+00	275	276
144	2	.42000E+02	.90000E+01	.00000E+00	277	278
145	2	.42000E+02	.12000E+02	.00000E+00	279	280

146	2	.43500E+02	.00000E+00	.00000E+00	281	282
147	2	.43500E+02	.30000E+01	.00000E+00	283	284
148	2	.43500E+02	.60000E+01	.00000E+00	285	286
149	2	.43500E+02	.90000E+01	.00000E+00	287	288
150	2	.43500E+02	.12000E+02	.00000E+00	289	290
151	2	.45000E+02	.00000E+00	.00000E+00	291	292
152	2	.45000E+02	.30000E+01	.00000E+00	293	294
153	2	.45000E+02	.60000E+01	.00000E+00	295	296
154	2	.45000E+02	.90000E+01	.00000E+00	297	298
155	2	.45000E+02	.12000E+02	.00000E+00	299	300
156	2	.46500E+02	.00000E+00	.00000E+00	301	302
157	2	.46500E+02	.30000E+01	.00000E+00	303	304
158	2	.46500E+02	.60000E+01	.00000E+00	305	306
159	2	.46500E+02	.90000E+01	.00000E+00	307	308
160	2	.46500E+02	.12000E+02	.00000E+00	309	310
161	2	.48000E+02	.00000E+00	.00000E+00	311	312
162	2	.48000E+02	.30000E+01	.00000E+00	313	314
163	2	.48000E+02	.60000E+01	.00000E+00	315	316
164	2	.48000E+02	.90000E+01	.00000E+00	317	318
165	2	.48000E+02	.12000E+02	.00000E+00	319	320

INPUT OF ELEMENT PROPERTIES (M= 0)
==================================

NUMBER OF GROUPS OF PROPERTIES (NGPE)= 1
NUMBER OF PROPERTIES PER GROUP (NPRE)= 3

CARDS OF ELEMENT PROPERTIES

))))) 1 .30000E+05 .25000E+00 .00000E+00
))))) 0 .00000E+00 .00000E+00 .00000E+00

INPUT OF ELEMENTS (M= 0)
===================

MAX. NUMBER OF ELEMENTS (NELT)= 64
MAX. NUMBER OF NODES PER ELEMENT (NNEL)= 6
DEFAULT ELEMENT TYPE (NTPE)= 3
NUMBER OF GROUPS OF ELEMENTS (NGRE)= 1
INDEX FOR NON SYMMETRIC PROBLEM (NSYM)= 0
INDEX FOR IDENTICAL ELEMENTS (NIDENT)= 0

ELEMENT: 1 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 1 6 11 12 13 7
ELEMENT: 2 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 11 16 21 22 23 17
ELEMENT: 3 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 21 26 31 32 33 27
ELEMENT: 4 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 31 36 41 42 43 37

```
ELEMENT:    5 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)   41   46   51   52   53   47
ELEMENT:    6 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)   51   56   61   62   63   57
ELEMENT:    7 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)   61   66   71   72   73   67
ELEMENT:    8 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)   71   76   81   82   83   77
ELEMENT:    9 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)   81   86   91   92   93   87
ELEMENT:   10 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)   91   96  101  102  103   97
ELEMENT:   11 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)  101  106  111  112  113  107
ELEMENT:   12 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)  111  116  121  122  123  117
ELEMENT:   13 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)  121  126  131  132  133  127
ELEMENT:   14 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)  131  136  141  142  143  137
ELEMENT:   15 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)  141  146  151  152  153  147
ELEMENT:   16 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)  151  156  161  162  163  157
ELEMENT:   17 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)    1    7   13    8    3    2
ELEMENT:   18 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)   11   17   23   18   13   12
ELEMENT:   19 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)   21   27   33   28   23   22
ELEMENT:   20 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)   31   37   43   38   33   32
ELEMENT:   21 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)   41   47   53   48   43   42
ELEMENT:   22 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)   51   57   63   58   53   52
ELEMENT:   23 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)   61   67   73   68   63   62
ELEMENT:   24 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)   71   77   83   78   73   72
ELEMENT:   25 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)   81   87   93   88   83   82
ELEMENT:   26 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)   91   97  103   98   93   92
ELEMENT:   27 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)  101  107  113  108  103  102
ELEMENT:   28 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)  111  117  123  118  113  112
ELEMENT:   29 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
    CONNECTIVITY (NE)  121  127  133  128  123  122
ELEMENT:   30 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
```

CONNECTIVITY (NE) 131 137 143 138 133 132
ELEMENT: 31 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 141 147 153 148 143 142
ELEMENT: 32 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 151 157 163 158 153 152
ELEMENT: 33 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 3 8 13 9 5 4
ELEMENT: 34 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 13 18 23 19 15 14
ELEMENT: 35 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 23 28 33 29 25 24
ELEMENT: 36 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 33 38 43 39 35 34
ELEMENT: 37 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 43 48 53 49 45 44
ELEMENT: 38 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 53 58 63 59 55 54
ELEMENT: 39 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 63 68 73 69 65 64
ELEMENT: 40 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 73 78 83 79 75 74
ELEMENT: 41 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 83 88 93 89 85 84
ELEMENT: 42 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 93 98 103 99 95 94
ELEMENT: 43 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 103 108 113 109 105 104
ELEMENT: 44 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 113 118 123 119 115 114
ELEMENT: 45 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 123 128 133 129 125 124
ELEMENT: 46 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 133 138 143 139 135 134
ELEMENT: 47 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 143 148 153 149 145 144
ELEMENT: 48 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 153 158 163 159 155 154
ELEMENT: 49 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 5 9 13 14 15 10
ELEMENT: 50 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 15 19 23 24 25 20
ELEMENT: 51 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 25 29 33 34 35 30
ELEMENT: 52 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 35 39 43 44 45 40
ELEMENT: 53 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 45 49 53 54 55 50
ELEMENT: 54 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 55 59 63 64 65 60
ELEMENT: 55 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP: 0 EL. PROP: 3 GROUP: 1
CONNECTIVITY (NE) 65 69 73 74 75 70

```
ELEMENT:   56 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
     CONNECTIVITY (NE)   75   79   83   84   85   80
ELEMENT:   57 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
     CONNECTIVITY (NE)   85   89   93   94   95   90
ELEMENT:   58 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
     CONNECTIVITY (NE)   95   99  103  104  105  100
ELEMENT:   59 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
     CONNECTIVITY (NE)  105  109  113  114  115  110
ELEMENT:   60 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
     CONNECTIVITY (NE)  115  119  123  124  125  120
ELEMENT:   61 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
     CONNECTIVITY (NE)  125  129  133  134  135  130
ELEMENT:   62 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
     CONNECTIVITY (NE)  135  139  143  144  145  140
ELEMENT:   63 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
     CONNECTIVITY (NE)  145  149  153  154  155  150
ELEMENT:   64 TYPE: 3 N.P.: 6 D.O.F.: 12 N. PROP:  0 EL. PROP:  3 GROUP:  1
     CONNECTIVITY (NE)  155  159  163  164  165  160

     MEAN BAND HEIGHT=    16.1 MAXIMUM=   25
     LENGTH OF A TRIANGLE IN KG         (NKG)=      5152
     NUMBER OF INTEGRATION POINTS       (NPG)=       192


INPUT OF CONCENTRADED LOADS (M= 0)
======================================


CARDS OF NODAL LOADS


>>>>>    1  .00000E+00 -.40000E+02
>>>>>  163    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
>>>>>    0  .00000E+00  .00000E+00


ASSEMBLING AND LINEAR SOLUTION (M= 0)
==============================
              INDEX FOR RESIDUAL COMPUTATION    (NRES)=    1
             ENERGY    (ENERG)=  .14252E+02

             ABSOLUTE VALUE OF MINIMUM PIVOT   =  .11094E+03 EQUATION:  320
                          ALGEBRAIC VALUE=  .11094E+03 EQUATION:  320
                          DETERMINANT      =  .16934E+09 * 10 **  1520


MAX. RESIDUAL VALUE=  .86544E-11 EQUATION  278


SOLUTION
```

NODES	X	Y	Z	DEGREES OF FREEDOM (* = PRESCRIBED)	
1	.00000E+00	.00000E+00	.00000E+00	.00000E+00 *	.00000E+00 *
2	.00000E+00	.30000E+01	.00000E+00	.00000E+00 *	.00000E+00 *
3	.00000E+00	.60000E+01	.00000E+00	.00000E+00 *	.00000E+00 *
4	.00000E+00	.90000E+01	.00000E+00	.00000E+00 *	.00000E+00 *
5	.00000E+00	.12000E+02	.00000E+00	.00000E+00 *	.00000E+00 *
6	.15000E+01	.00000E+00	.00000E+00	-.40761E-02	-.18132E-02
7	.15000E+01	.30000E+01	.00000E+00	-.17307E-02	-.75535E-03
8	.15000E+01	.60000E+01	.00000E+00	.13830E-15	-.56466E-03
9	.15000E+01	.90000E+01	.00000E+00	.17307E-02	-.75535E-03
10	.15000E+01	.12000E+02	.00000E+00	.40761E-02	-.18132E-02
11	.30000E+01	.00000E+00	.00000E+00	-.79275E-02	-.40486E-02
12	.30000E+01	.30000E+01	.00000E+00	-.35972E-02	-.26875E-02
13	.30000E+01	.60000E+01	.00000E+00	.28364E-16	-.23173E-02
14	.30000E+01	.90000E+01	.00000E+00	.35972E-02	-.26875E-02
15	.30000E+01	.12000E+02	.00000E+00	.79275E-02	-.40486E-02
16	.45000E+01	.00000E+00	.00000E+00	-.11572E-01	-.69088E-02
17	.45000E+01	.30000E+01	.00000E+00	-.54536E-02	-.55596E-02
18	.45000E+01	.60000E+01	.00000E+00	.43342E-16	-.51172E-02
19	.45000E+01	.90000E+01	.00000E+00	.54536E-02	-.55596E-02
20	.45000E+01	.12000E+02	.00000E+00	.11572E-01	-.69088E-02
21	.60000E+01	.00000E+00	.00000E+00	-.15119E-01	-.10632E-01
22	.60000E+01	.30000E+01	.00000E+00	-.72461E-02	-.93194E-02
23	.60000E+01	.60000E+01	.00000E+00	.58517E-16	-.88742E-02
24	.60000E+01	.90000E+01	.00000E+00	.72461E-02	-.93194E-02
25	.60000E+01	.12000E+02	.00000E+00	.15119E-01	-.10632E-01
26	.75000E+01	.00000E+00	.00000E+00	-.18534E-01	-.15220E-01
27	.75000E+01	.30000E+01	.00000E+00	-.89728E-02	-.13951E-01
28	.75000E+01	.60000E+01	.00000E+00	.73952E-16	-.13515E-01
29	.75000E+01	.90000E+01	.00000E+00	.89728E-02	-.13951E-01
30	.75000E+01	.12000E+02	.00000E+00	.18534E-01	-.15220E-01
31	.90000E+01	.00000E+00	.00000E+00	-.21848E-01	-.20637E-01
32	.90000E+01	.30000E+01	.00000E+00	-.10633E-01	-.19421E-01
33	.90000E+01	.60000E+01	.00000E+00	.90018E-16	-.19015E-01
34	.90000E+01	.90000E+01	.00000E+00	.10633E-01	-.19421E-01
35	.90000E+01	.12000E+02	.00000E+00	.21848E-01	-.20637E-01
36	.10500E+02	.00000E+00	.00000E+00	-.25022E-01	-.26879E-01
37	.10500E+02	.30000E+01	.00000E+00	-.12229E-01	-.25704E-01
38	.10500E+02	.60000E+01	.00000E+00	.10632E-15	-.25306E-01
39	.10500E+02	.90000E+01	.00000E+00	.12229E-01	-.25704E-01
40	.10500E+02	.12000E+02	.00000E+00	.25022E-01	-.26879E-01
41	.12000E+02	.00000E+00	.00000E+00	-.28093E-01	-.33888E-01
42	.12000E+02	.30000E+01	.00000E+00	-.13761E-01	-.32766E-01
43	.12000E+02	.60000E+01	.00000E+00	.12231E-15	-.32397E-01
44	.12000E+02	.90000E+01	.00000E+00	.13761E-01	-.32766E-01
45	.12000E+02	.12000E+02	.00000E+00	.28093E-01	-.33888E-01
46	.13500E+02	.00000E+00	.00000E+00	-.31019E-01	-.41661E-01

47	.13500E+02	.30000E+01	.00000E+00	-.15230E-01	-.40581E-01
48	.13500E+02	.60000E+01	.00000E+00	.14188E-15	-.40216E-01
49	.13500E+02	.90000E+01	.00000E+00	.15230E-01	-.40581E-01
50	.13500E+02	.12000E+02	.00000E+00	.31019E-01	-.41651E-01
51	.15000E+02	.00000E+00	.00000E+00	-.33842E-01	-.50138E-01
52	.15000E+02	.30000E+01	.00000E+00	-.16637E-01	-.49111E-01
53	.15000E+02	.60000E+01	.00000E+00	.16098E-15	-.48774E-01
54	.15000E+02	.90000E+01	.00000E+00	.16637E-01	-.49111E-01
55	.15000E+02	.12000E+02	.00000E+00	.33842E-01	-.50138E-01
56	.16500E+02	.00000E+00	.00000E+00	-.36519E-01	-.59318E-01
57	.16500E+02	.30000E+01	.00000E+00	-.17981E-01	-.58332E-01
58	.16500E+02	.60000E+01	.00000E+00	.18089E-15	-.57999E-01
59	.16500E+02	.90000E+01	.00000E+00	.17981E-01	-.58332E-01
60	.16500E+02	.12000E+02	.00000E+00	.36519E-01	-.59318E-01
61	.18000E+02	.00000E+00	.00000E+00	-.39092E-01	-.69139E-01
62	.18000E+02	.30000E+01	.00000E+00	-.19262E-01	-.68206E-01
63	.18000E+02	.60000E+01	.00000E+00	.20285E-15	-.67900E-01
64	.18000E+02	.90000E+01	.00000E+00	.19262E-01	-.68206E-01
65	.18000E+02	.12000E+02	.00000E+00	.39092E-01	-.69139E-01
66	.19500E+02	.00000E+00	.00000E+00	-.41519E-01	-.79599E-01
67	.19500E+02	.30000E+01	.00000E+00	-.20481E-01	-.78707E-01
68	.19500E+02	.60000E+01	.00000E+00	.22348E-15	-.78406E-01
69	.19500E+02	.90000E+01	.00000E+00	.20481E-01	-.78707E-01
70	.19500E+02	.12000E+02	.00000E+00	.41519E-01	-.79599E-01
71	.21000E+02	.00000E+00	.00000E+00	-.43842E-01	-.90639E-01
72	.21000E+02	.30000E+01	.00000E+00	-.21637E-01	-.89800E-01
73	.21000E+02	.60000E+01	.00000E+00	.24516E-15	-.89526E-01
74	.21000E+02	.90000E+01	.00000E+00	.21637E-01	-.89800E-01
75	.21000E+02	.12000E+02	.00000E+00	.43842E-01	-.90639E-01
76	.22500E+02	.00000E+00	.00000E+00	-.46019E-01	-.10226E+00
77	.22500E+02	.30000E+01	.00000E+00	-.22731E-01	-.10146E+00
78	.22500E+02	.60000E+01	.00000E+00	.25923E-15	-.10119E+00
79	.22500E+02	.90000E+01	.00000E+00	.22731E-01	-.10146E+00
80	.22500E+02	.12000E+02	.00000E+00	.46019E-01	-.10226E+00
81	.24000E+02	.00000E+00	.00000E+00	-.48092E-01	-.11439E+00
82	.24000E+02	.30000E+01	.00000E+00	-.23762E-01	-.11364E+00
83	.24000E+02	.60000E+01	.00000E+00	.26824E-15	-.11340E+00
84	.24000E+02	.90000E+01	.00000E+00	.23762E-01	-.11364E+00
85	.24000E+02	.12000E+02	.00000E+00	.48092E-01	-.11439E+00
86	.25500E+02	.00000E+00	.00000E+00	-.50019E-01	-.12704E+00
87	.25500E+02	.30000E+01	.00000E+00	-.24731E-01	-.12633E+00
88	.25500E+02	.60000E+01	.00000E+00	.29322E-15	-.12609E+00
89	.25500E+02	.90000E+01	.00000E+00	.24731E-01	-.12633E+00
90	.25500E+02	.12000E+02	.00000E+00	.50019E-01	-.12704E+00
91	.27000E+02	.00000E+00	.00000E+00	-.51842E-01	-.14014E+00
92	.27000E+02	.30000E+01	.00000E+00	-.25637E-01	-.13949E+00
93	.27000E+02	.60000E+01	.00000E+00	.32138E-15	-.13928E+00
94	.27000E+02	.90000E+01	.00000E+00	.25637E-01	-.13949E+00
95	.27000E+02	.12000E+02	.00000E+00	.51842E-01	-.14014E+00
96	.28500E+02	.00000E+00	.00000E+00	-.53519E-01	-.15370E+00
97	.28500E+02	.30000E+01	.00000E+00	-.26481E-01	-.15308E+00

98	.28500E+02	.60000E+01	.00000E+00	.34012E-15	-.15288E+00
99	.28500E+02	.90000E+01	.00000E+00	.26481E-01	-.15308E+00
100	.28500E+02	.12000E+02	.00000E+00	.53519E-01	-.15370E+00
101	.30000E+02	.00000E+00	.00000E+00	-.55092E-01	-.16764E+00
102	.30000E+02	.30000E+01	.00000E+00	-.27262E-01	-.16708E+00
103	.30000E+02	.60000E+01	.00000E+00	.35740E-15	-.16690E+00
104	.30000E+02	.90000E+01	.00000E+00	.27262E-01	-.16708E+00
105	.30000E+02	.12000E+02	.00000E+00	.55092E-01	-.16764E+00
106	.31500E+02	.00000E+00	.00000E+00	-.56518E-01	-.18198E+00
107	.31500E+02	.30000E+01	.00000E+00	-.27981E-01	-.18146E+00
108	.31500E+02	.60000E+01	.00000E+00	.38587E-15	-.18128E+00
109	.31500E+02	.90000E+01	.00000E+00	.27981E-01	-.18146E+00
110	.31500E+02	.12000E+02	.00000E+00	.56518E-01	-.18198E+00
111	.33000E+02	.00000E+00	.00000E+00	-.57841E-01	-.19664E+00
112	.33000E+02	.30000E+01	.00000E+00	-.28637E-01	-.19618E+00
113	.33000E+02	.60000E+01	.00000E+00	.41094E-15	-.19603E+00
114	.33000E+02	.90000E+01	.00000E+00	.28637E-01	-.19618E+00
115	.33000E+02	.12000E+02	.00000E+00	.57841E-01	-.19664E+00
116	.34500E+02	.00000E+00	.00000E+00	-.59017E-01	-.21163E+00
117	.34500E+02	.30000E+01	.00000E+00	-.29232E-01	-.21121E+00
118	.34500E+02	.60000E+01	.00000E+00	.42791E-15	-.21106E+00
119	.34500E+02	.90000E+01	.00000E+00	.29232E-01	-.21121E+00
120	.34500E+02	.12000E+02	.00000E+00	.59017E-01	-.21163E+00
121	.36000E+02	.00000E+00	.00000E+00	-.60090E-01	-.22689E+00
122	.36000E+02	.30000E+01	.00000E+00	-.29764E-01	-.22652E+00
123	.36000E+02	.60000E+01	.00000E+00	.44969E-15	-.22640E+00
124	.36000E+02	.90000E+01	.00000E+00	.29764E-01	-.22652E+00
125	.36000E+02	.12000E+02	.00000E+00	.60090E-01	-.22689E+00
126	.37500E+02	.00000E+00	.00000E+00	-.61013E-01	-.24242E+00
127	.37500E+02	.30000E+01	.00000E+00	-.30234E-01	-.24208E+00
128	.37500E+02	.60000E+01	.00000E+00	.46387E-15	-.24197E+00
129	.37500E+02	.90000E+01	.00000E+00	.30234E-01	-.24208E+00
130	.37500E+02	.12000E+02	.00000E+00	.61013E-01	-.24242E+00
131	.39000E+02	.00000E+00	.00000E+00	-.61831E-01	-.25815E+00
132	.39000E+02	.30000E+01	.00000E+00	-.30645E-01	-.25786E+00
133	.39000E+02	.60000E+01	.00000E+00	.47201E-15	-.25777E+00
134	.39000E+02	.90000E+01	.00000E+00	.30645E-01	-.25786E+00
135	.39000E+02	.12000E+02	.00000E+00	.61831E-01	-.25815E+00
136	.40500E+02	.00000E+00	.00000E+00	-.62492E-01	-.27408E+00
137	.40500E+02	.30000E+01	.00000E+00	-.30997E-01	-.27383E+00
138	.40500E+02	.60000E+01	.00000E+00	.48161E-15	-.27373E+00
139	.40500E+02	.90000E+01	.00000E+00	.30997E-01	-.27383E+00
140	.40500E+02	.12000E+02	.00000E+00	.62492E-01	-.27408E+00
141	.42000E+02	.00000E+00	.00000E+00	-.63044E-01	-.29016E+00
142	.42000E+02	.30000E+01	.00000E+00	-.31295E-01	-.28994E+00
143	.42000E+02	.60000E+01	.00000E+00	.48999E-15	-.28986E+00
144	.42000E+02	.90000E+01	.00000E+00	.31295E-01	-.28994E+00
145	.42000E+02	.12000E+02	.00000E+00	.63044E-01	-.29016E+00
146	.43500E+02	.00000E+00	.00000E+00	-.63411E-01	-.30634E+00
147	.43500E+02	.30000E+01	.00000E+00	-.31544E-01	-.30618E+00
148	.43500E+02	.60000E+01	.00000E+00	.54901E-15	-.30610E+00

149	.43500E+02	.90000E+01	.00000E+00	.31544E-01	-.30618E+00
150	.43500E+02	.12000E+02	.00000E+00	.63411E-01	-.30634E+00
151	.45000E+02	.00000E+00	.00000E+00	-.63649E-01	-.32257E+00
152	.45000E+02	.30000E+01	.00000E+00	-.31750E-01	-.32249E+00
153	.45000E+02	.60000E+01	.00000E+00	.61189E-15	-.32244E+00
154	.45000E+02	.90000E+01	.00000E+00	.31750E-01	-.32249E+00
155	.45000E+02	.12000E+02	.00000E+00	.63649E-01	-.32257E+00
156	.46500E+02	.00000E+00	.00000E+00	-.63813E-01	-.33873E+00
157	.46500E+02	.30000E+01	.00000E+00	-.31878E-01	-.33884E+00
158	.46500E+02	.60000E+01	.00000E+00	.65434E-15	-.33912E+00
159	.46500E+02	.90000E+01	.00000E+00	.31878E-01	-.33884E+00
160	.46500E+02	.12000E+02	.00000E+00	.63813E-01	-.33873E+00
161	.48000E+02	.00000E+00	.00000E+00	-.63868E-01	-.35477E+00
162	.48000E+02	.30000E+01	.00000E+00	-.31823E-01	-.35507E+00
163	.48000E+02	.60000E+01	.00000E+00	.66939E-15	-.35630E+00
164	.48000E+02	.90000E+01	.00000E+00	.31823E-01	-.35507E+00
165	.48000E+02	.12000E+02	.00000E+00	.63868E-01	-.35477E+00

CONTAINTES DANS L ELEMENT 1

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.15000E+01	.00000E+00	-.26425E-02	.51777E-03	-.44555E-03	-.80418E+02	-.45714E+01	-.53467E+01	-86.0
						-.41963E+01	-.80793E+02	.38299E+02	
2	.30000E+01	.30000E+01	-.11694E-02	.28854E-03	-.10756E-03	-.35114E+02	-.12208E+00	-.12907E+01	-87.9
						-.74532E-01	-.35161E+02	.17543E+02	
3	.15000E+01	.30000E+01	-.13192E-02	.18747E-03	-.48780E-03	-.40714E+02	-.45543E+01	-.58536E+01	-81.0
						-.36303E+01	-.41638E+02	.19004E+02	

CONTAINTES DANS L ELEMENT 2

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.45000E+01	.00000E+00	-.23971E-02	.59440E-03	-.50558E-04	-.71953E+02	-.15626E+00	-.60663E+00	-89.5
						-.15113E+00	-.71958E+02	.35903E+02	
2	.60000E+01	.30000E+01	-.11622E-02	.29304E-03	-.27446E-03	-.34847E+02	.79468E-01	-.32935E+01	-84.7
						.38732E+00	-.35155E+02	.17771E+02	
3	.45000E+01	.30000E+01	-.12279E-02	.30507E-03	-.28349E-03	-.36851E+02	-.60618E-01	-.34019E+01	-84.8
						.25130E+00	-.37163E+02	.18707E+02	

CONTAINTES DANS L ELEMENT 3

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.75000E+01	.00000E+00	-.22432E-02	.55844E-03	-.50915E-04	-.67314E+02	-.75302E-01	-.61098E+00	-89.5
						-.69751E-01	-.67319E+02	.33625E+02	
2	.90000E+01	.30000E+01	-.10735E-02	.27037E-03	-.28154E-03	-.32188E+02	.64231E-01	-.33785E+01	-84.1
						.41434E+00	-.32538E+02	.16476E+02	

3	.75000E+01	.30000E+01	-.11400E-02	.28810E-03	-.28056E-03	-.34177E+02	.98918E-01	-.33667E+01	-84.4
						.42648E+00	-.34504E+02	.17465E+02	

CONTAINTES DANS L ELEMENT 4

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.10500E+02	.00000E+00	-.20817E-02	.51677E-03	-.57160E-04	-.62479E+02	-.11664E+00	-.68592E+00	-89.4
						-.10910E+00	-.62486E+02	.31189E+02	
2	.12000E+02	.30000E+01	-.98726E-03	.24843E-03	-.28138E-03	-.29605E+02	.51747E-01	-.33766E+01	-83.6
						.43134E+00	-.29984E+02	.15208E+02	
3	.10500E+02	.30000E+01	-.10555E-02	.26621E-03	-.28314E-03	-.31647E+02	.74499E-01	-.33977E+01	-84.0
						.43434E+00	-.32006E+02	.16220E+02	

CONTAINTES DANS L ELEMENT 5

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.13500E+02	.00000E+00	-.19163E-02	.47498E-03	-.59055E-04	-.57522E+02	-.13088E+00	-.70866E+00	-89.3
						-.12213E+00	-.57530E+02	.28704E+02	
2	.15000E+02	.30000E+01	-.90316E-03	.22730E-03	-.28119E-03	-.27083E+02	.48224E-01	-.33743E+01	-83.0
						.46159E+00	-.27496E+02	.13979E+02	
3	.13500E+02	.30000E+01	-.97190E-03	.24497E-03	-.28399E-03	-.29141E+02	.63777E-01	-.34078E+01	-83.4
						.45616E+00	-.29533E+02	.14995E+02	

CONTAINTES DANS L ELEMENT 6

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.16500E+02	.00000E+00	-.17499E-02	.43328E-03	-.59506E-04	-.52531E+02	-.13437E+00	-.71407E+00	-89.2
						-.12464E+00	-.52541E+02	.26208E+02	
2	.18000E+02	.30000E+01	-.81965E-03	.20640E-03	-.28114E-03	-.24578E+02	.47407E-01	-.33736E+01	-82.3
						.50123E+00	-.25032E+02	.12766E+02	
3	.16500E+02	.30000E+01	-.88851E-03	.22403E-03	-.28419E-03	-.26640E+02	.61003E-01	-.34103E+01	-82.8
						.48969E+00	-.27069E+02	.13779E+02	

CONTAINTES DANS L ELEMENT 7

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.19500E+02	.00000E+00	-.15833E-02	.39161E-03	-.59604E-04	-.47533E+02	-.13514E+00	-.71525E+00	-89.1
						-.12435E+00	-.47544E+02	.23710E+02	
2	.21000E+02	.30000E+01	-.73628E-03	.18555E-03	-.28112E-03	-.22077E+02	.47236E-01	-.33735E+01	-81.5
						.55019E+00	-.22580E+02	.11565E+02	
3	.19500E+02	.30000E+01	-.80516E-03	.20318E-03	-.28424E-03	-.24140E+02	.60378E-01	-.34109E+01	-82.1
						.53193E+00	-.24611E+02	.12572E+02	

CONTRAINTES DANS L ELEMENT 8

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.22500E+02	.00000E+00	-.14167E-02	.34994E-03	-.59628E-04	-.42534E+02	-.13527E+00	-.71554E+00	-89.0
						-.12320E+00	-.42546E+02	.21211E+02	
2	.24000E+02	.30000E+01	-.65294E-03	.16471E-03	-.28112E-03	-.19577E+02	.47227E-01	-.33734E+01	-80.5
						.61095E+00	-.20140E+02	.10376E+02	
3	.22500E+02	.30000E+01	-.72183E-03	.18234E-03	-.28425E-03	-.21640E+02	.60248E-01	-.34110E+01	-81.3
						.58378E+00	-.22163E+02	.11374E+02	

CONTRAINTES DANS L ELEMENT 9

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.25500E+02	.00000E+00	-.12500E-02	.30827E-03	-.59647E-04	-.37534E+02	-.13516E+00	-.71576E+00	-88.9
						-.12146E+00	-.37547E+02	.18713E+02	
2	.27000E+02	.30000E+01	-.56961E-03	.14388E-03	-.28112E-03	-.17077E+02	.47355E-01	-.33735E+01	-79.2
						.68797E+00	-.17717E+02	.92026E+01	
3	.25500E+02	.30000E+01	-.63850E-03	.16151E-03	-.28424E-03	-.19140E+02	.60238E-01	-.34109E+01	-80.2
						.64818E+00	-.19728E+02	.10188E+02	

CONTRAINTES DANS L ELEMENT 10

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.28500E+02	.00000E+00	-.10833E-02	.26663E-03	-.59718E-04	-.32533E+02	-.13443E+00	-.71662E+00	-85.7
						-.11858E+00	-.32549E+02	.16215E+02	
2	.30000E+02	.30000E+01	-.48630E-03	.12308E-03	-.28113E-03	-.14577E+02	.48011E-01	-.33735E+01	-77.6
						.78867E+00	-.15318E+02	.80531E+01	
3	.28500E+02	.30000E+01	-.55519E-03	.14068E-03	-.28422E-03	-.16641E+02	.60316E-01	-.34106E+01	-78.9
						.72996E+00	-.17310E+02	.90201E+01	

CONTRAINTES DANS L ELEMENT 11

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.31500E+02	.00000E+00	-.91654E-03	.22504E-03	-.60055E-04	-.27529E+02	-.13093E+00	-.72066E+00	-88.5
						-.11199E+00	-.27548E+02	.13718E+02	
2	.33000E+02	.30000E+01	-.40306E-03	.10236E-03	-.28116E-03	-.12079E+02	.51108E-01	-.33739E+01	-75.5
						.92639E+00	-.12954E+02	.69403E+01	
3	.31500E+02	.30000E+01	-.47197E-03	.11989E-03	-.28409E-03	-.14144E+02	.60720E-01	-.34090E+01	-77.2
						.83650E+00	-.14920E+02	.78780E+01	

CONTRAINTES DANS L ELEMENT 12

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.34500E+02	.00000E+00	-.74941E-03	.18376E-03	-.61633E-04	-.22511E+02	-.11486E+00	-.73960E+00	-88.1
						-.90463E-01	-.22535E+02	.11222E+02	
2	.36000E+02	.30000E+01	-.32015E-03	.82072E-04	-.28130E-03	-.95884E+01	.65076E-01	-.33756E+01	-72.5
						.11284E+01	-.10652E+02	.58900E+01	
3	.34500E+02	.30000E+01	-.38916E-03	.99246E-04	-.28348E-03	-.11659E+02	.62603E-01	-.34018E+01	-74.9
						.97830E+00	-.12575E+02	.67765E+01	

CONTRAINTES DANS L ELEMENT 13

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.37500E+02	.00000E+00	-.58057E-03	.14368E-03	-.68621E-04	-.17429E+02	-.46786E-01	-.82345E+00	-87.3
						-.78630E-02	-.17468E+02	.87299E+01	
2	.39000E+02	.30000E+01	-.23872E-03	.63479E-04	-.28189E-03	-.71313E+01	.12156E+00	-.33827E+01	-68.5
						.14543E+01	-.84640E+01	.49592E+01	
3	.37500E+02	.30000E+01	-.30820E-03	.79265E-04	-.28089E-03	-.92283E+01	.70860E-01	-.33706E+01	-72.0
						.11641E+01	-.10322E+02	.57428E+01	

CONTRAINTES DANS L ELEMENT 14

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.40500E+02	.00000E+00	-.40429E-03	.10677E-03	-.95552E-04	-.12083E+02	.18232E+00	-.11466E+01	-84.7
						.28859E+00	-.12190E+02	.62391E+01	
2	.42000E+02	.30000E+01	-.16265E-03	.49449E-04	-.28351E-03	-.48092E+01	.28117E+00	-.34033E+01	-63.4
						.19858E+01	-.65138E+01	.42498E+01	
3	.40500E+02	.30000E+01	-.23462E-03	.61830E-04	-.27178E-03	-.70132E+01	.10162E+00	-.32614E+01	-68.7
						.13704E+01	-.82819E+01	.48261E+01	

CONTRAINTES DANS L ELEMENT 15

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.43500E+02	.00000E+00	-.20152E-03	.60935E-04	-.15722E-03	-.59610E+01	.33779E+00	-.18866E+01	-74.5
						.85964E+00	-.64829E+01	.36713E+01	
2	.45000E+02	.30000E+01	-.94911E-04	.21937E-04	-.28073E-03	-.28617E+01	-.57312E-01	-.33688E+01	-56.3
						.21894E+01	-.51084E+01	.36489E+01	
3	.43500E+02	.30000E+01	-.18016E-03	.49195E-04	-.26112E-03	-.53717E+01	.13292E+00	-.31334E+01	-65.6
						.15512E+01	-.67899E+01	.41705E+01	

CONTRAINTES DANS L ELEMENT 16

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.46500E+02	.00000E+00	-.73112E-04	.11770E-03	-.50210E-04	-.13979E+01	.31816E+01	-.60253E+00	-82.6

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
						.32596E+01	-.14759E+01	.23677E+01	
2	.48000E+02	.30000E+01	.73112E-04	-.25479E-03	-.13257E-03	.30126E+00	-.75684E+01	-.15908E+01	-11.0
						.61068E+00	-.78778E+01	.42442E+01	
3	.46500E+02	.30000E+01	.22904E-17	-.19011E-03	-.25348E-03	-.15209E+01	-.60834E+01	-.30417E+01	-26.6
						.13145E-12	-.76043E+01	.38021E+01	

CONTRAINTES DANS L ELEMENT 17

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.15000E+01	.30000E+01	-.11538E-02	.63565E-04	-.32266E-03	-.36414E+02	-.71965E+01	-.38719E+01	-82.6
						-.66921E+01	-.36918E+02	.15113E+02	
2	.15000E+01	.60000E+01	.94548E-17	.63565E-04	-.19553E-03	.50852E+00	.20341E+01	-.23464E+01	-54.0
						.37386E+01	-.11959E+01	.24672E+01	
3	.00000E+00	.30000E+01	-.11538E-02	.00000E+00	-.10756E-03	-.36922E+02	-.92306E+01	-.12907E+01	-87.3
						-.91706E+01	-.36982E+02	.13906E+02	

CONTRAINTES DANS L ELEMENT 18

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.45000E+01	.30000E+01	-.12376E-02	.31262E-03	-.29373E-03	-.37102E+02	.10321E+00	-.35248E+01	-84.6
						.43421E+00	-.37433E+02	.18934E+02	
2	.45000E+01	.60000E+01	.10051E-16	-.17682E-04	-.48997E-03	-.14146E+00	-.56584E+00	-.58796E+01	-44.0
						.55298E+01	-.62371E+01	.58834E+01	
3	.30000E+01	.30000E+01	-.12376E-02	.28854E-03	-.27446E-03	-.37294E+02	-.66731E+00	-.32935E+01	-84.9
						-.37352E+00	-.37588E+02	.18607E+02	

CONTRAINTES DANS L ELEMENT 19

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.75000E+01	.30000E+01	-.11511E-02	.28975E-03	-.27833E-03	-.34517E+02	.63288E-01	-.33400E+01	-84.5
						.38293E+00	-.34837E+02	.17610E+02	
2	.75000E+01	.60000E+01	.10500E-16	.43041E-06	-.49378E-03	.34433E-02	.13773E-01	-.59253E+01	-45.0
						.59339E+01	-.59167E+01	.59253E+01	
3	.60000E+01	.30000E+01	-.11511E-02	.29304E-03	-.28154E-03	-.34491E+02	.16850E+00	-.33785E+01	-84.5
						.49476E+00	-.34817E+02	.17656E+02	

CONTRAINTES DANS L ELEMENT 20

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.10500E+02	.30000E+01	-.10640E-02	.26786E-03	-.28220E-03	-.31906E+02	.59492E-01	-.33864E+01	-84.0
						.41431E+00	-.32260E+02	.16337E+02	
2	.10500E+02	.60000E+01	.10764E-16	-.24726E-05	-.48140E-03	-.19781E-01	-.79124E-01	-.57768E+01	-44.9
						.57274E+01	-.58263E+01	.57769E+01	

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
3	.90000E+01	.30000E+01	-.10540E-02	.27037E-03	-.28138E-03	-.31886E+02	.13977E+00	-.33766E+01	-84.0
						.49191E+00	-.32238E+02	.16365E+02	

CONTAINTES DANS L ELEMENT 21

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.13500E+02	.30000E+01	-.97960E-03	.24677E-03	-.28375E-03	-.29373E+02	.59921E-01	-.34050E+01	-83.5
						.44869E+00	-.29752E+02	.15105E+02	
2	.13500E+02	.60000E+01	.12890E-16	-.37887E-05	-.47749E-03	-.30310E-01	-.12124E+00	-.57299E+01	-44.8
						.56543E+01	-.58059E+01	.57301E+01	
3	.12000E+02	.30000E+01	-.97960E-03	.24843E-03	-.28119E-03	-.29360E+02	.11298E+00	-.33743E+01	-83.6
						.49436E+00	-.29741E+02	.15118E+02	

CONTAINTES DANS L ELEMENT 22

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.16500E+02	.30000E+01	-.89602E-03	.22588E-03	-.28414E-03	-.26866E+02	.60136E-01	-.34097E+01	-82.9
						.48520E+00	-.27291E+02	.13888E+02	
2	.16500E+02	.60000E+01	.13957E-16	-.41315E-05	-.47656E-03	-.33052E-01	-.13221E+00	-.57187E+01	-44.8
						.56363E+01	-.58015E+01	.57189E+01	
3	.15000E+02	.30000E+01	-.89602E-03	.22730E-03	-.28114E-03	-.26854E+02	.10537E+00	-.33736E+01	-83.0
						.52112E+00	-.27270E+02	.13896E+02	

CONTAINTES DANS L ELEMENT 23

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.19500E+02	.30000E+01	-.81263E-03	.20504E-03	-.28423E-03	-.24364E+02	.60195E-01	-.34107E+01	-82.2
						.52755E+00	-.24831E+02	.12679E+02	
2	.19500E+02	.60000E+01	.14105E-16	-.42089E-05	-.47635E-03	-.33671E-01	-.13468E+00	-.57162E+01	-44.7
						.56323E+01	-.58006E+01	.57164E+01	
3	.18000E+02	.30000E+01	-.81263E-03	.20640E-03	-.28112E-03	-.24353E+02	.10360E+00	-.33735E+01	-82.3
						.56039E+00	-.24810E+02	.12685E+02	

CONTAINTES DANS L ELEMENT 24

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.22500E+02	.30000E+01	-.72929E-03	.18420E-03	-.28424E-03	-.21864E+02	.60219E-01	-.34109E+01	-81.4
						.57864E+00	-.22382E+02	.11480E+02	
2	.22500E+02	.60000E+01	.76941E-17	-.42250E-05	-.47631E-03	-.33800E-01	-.13520E+00	-.57157E+01	-44.7
						.56314E+01	-.58004E+01	.57159E+01	
3	.21000E+02	.30000E+01	-.72929E-03	.18555E-03	-.28112E-03	-.21853E+02	.10322E+00	-.33734E+01	-81.5
						.60984E+00	-.22359E+02	.11485E+02	

CONTRAINTES DANS L ELEMENT 25

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.25500E+02	.30000E+01	-.64595E-03	.16337E-03	-.28425E-03	-.19364E+02	.60277E-01	-.34109E+01	-80.3
						.64185E+00	-.19945E+02	.10293E+02	
2	.25500E+02	.60000E+01	.17713E-16	-.42267E-05	-.47629E-03	-.33814E-01	-.13525E+00	-.57155E+01	-44.7
						.56312E+01	-.58002E+01	.57157E+01	
3	.24000E+02	.30000E+01	-.64595E-03	.16471E-03	-.28112E-03	-.19353E+02	.10316E+00	-.33735E+01	-80.4
						.67148E+00	-.19921E+02	.10296E+02	

CONTRAINTES DANS L ELEMENT 26

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.28500E+02	.30000E+01	-.56263E-03	.14255E-03	-.28423E-03	-.16864E+02	.60545E-01	-.34108E+01	-79.0
						.72206E+00	-.17525E+02	.91237E+01	
2	.28500E+02	.60000E+01	.12006E-16	-.42187E-05	-.47625E-03	-.33749E-01	-.13500E+00	-.57150E+01	-44.7
						.56308E+01	-.57996E+01	.57152E+01	
3	.27000E+02	.30000E+01	-.56263E-03	.14388E-03	-.28113E-03	-.16853E+02	.10323E+00	-.33735E+01	-79.2
						.74976E+00	-.17500E+02	.91247E+01	

CONTRAINTES DANS L ELEMENT 27

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.31500E+02	.30000E+01	-.47934E-03	.12177E-03	-.28415E-03	-.14365E+02	.61815E-01	-.34098E+01	-77.3
						.82716E+00	-.15130E+02	.79786E+01	
2	.31500E+02	.60000E+01	.17847E-16	-.41771E-05	-.47605E-03	-.33417E-01	-.13367E+00	-.57126E+01	-44.7
						.56293E+01	-.57964E+01	.57129E+01	
3	.30000E+02	.30000E+01	-.47934E-03	.12308E-03	-.28116E-03	-.14354E+02	.10367E+00	-.33739E+01	-77.5
						.85224E+00	-.15103E+02	.79776E+01	

CONTRAINTES DANS L ELEMENT 28

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.34500E+02	.30000E+01	-.39623E-03	.10117E-03	-.28380E-03	-.11870E+02	.67654E-01	-.34056E+01	-75.1
						.97085E+00	-.12773E+02	.68720E+01	
2	.34500E+02	.60000E+01	.12916E-16	-.39840E-05	-.47515E-03	-.31872E-01	-.12749E+00	-.57018E+01	-44.8
						.56223E+01	-.57817E+01	.57020E+01	
3	.33000E+02	.30000E+01	-.39623E-03	.10236E-03	-.28130E-03	-.11860E+02	.10573E+00	-.33756E+01	-75.3
						.99230E+00	-.12747E+02	.68697E+01	

CONTRAINTES DANS L ELEMENT 29

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA

						SIG1	SIG2	TAUMAX	
1	.37500E+02	.30000E+01	-.31391E-03	.81368E-04	-.28224E-03	-.93943E+01	.92467E-01	-.33868E+01	-72.2
						.11775E+01	-.10479E+02	.58284E+01	
2	.37500E+02	.60000E+01	.74417E-17	-.31501E-05	-.47115E-03	-.25201E-01	-.10080E+00	-.56538E+01	-44.8
						.55909E+01	-.57169E+01	.56539E+01	
3	.36000E+02	.30000E+01	-.31391E-03	.82072E-04	-.28189E-03	-.93886E+01	.11501E+00	-.33827E+01	-72.3
						.11961E+01	-.10470E+02	.58329E+01	

CONTAINTES DANS L ELEMENT - 30

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.40500E+02	.30000E+01	-.23489E-03	.64245E-04	-.27647E-03	-.70025E+01	.17672E+00	-.33177E+01	-68.5
						.14751E+01	-.83009E+01	.48880E+01	
2	.40500E+02	.60000E+01	.59905E-17	-.17029E-06	-.45564E-03	-.13623E-02	-.54493E-02	-.54677E+01	-45.0
						.54642E+01	-.54711E+01	.54677E+01	
3	.39000E+02	.30000E+01	-.23489E-03	.63479E-04	-.28361E-03	-.70086E+01	.15223E+00	-.34033E+01	-68.2
						.15117E+01	-.83680E+01	.49398E+01	

CONTAINTES DANS L ELEMENT 31

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.43500E+02	.30000E+01	-.16606E-03	.48477E-04	-.26673E-03	-.49262E+01	.22277E+00	-.32008E+01	-64.4
						.17560E+01	-.64594E+01	.41077E+01	
2	.43500E+02	.60000E+01	.40635E-16	.35364E-05	-.42014E-03	.28291E-01	.11316E+00	-.50417E+01	-45.2
						.51126E+01	-.49711E+01	.50419E+01	
3	.42000E+02	.30000E+01	-.16606E-03	.49449E-04	-.28073E-03	-.49184E+01	.25387E+00	-.33688E+01	-63.8
						.19147E+01	-.65792E+01	.42469E+01	

CONTAINTES DANS L ELEMENT 32

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.46500E+02	.30000E+01	-.85147E-04	-.88824E-04	-.41347E-03	-.34353E+01	-.35235E+01	-.49616E+01	-44.7
						.14824E+01	-.84413E+01	.49618E+01	
2	.46500E+02	.60000E+01	.19166E-16	-.10056E-03	-.68437E-03	-.80452E+00	-.32181E+01	-.82125E+01	-40.8
						.62894E+01	-.10312E+02	.83007E+01	
3	.45000E+02	.30000E+01	-.85147E-04	.21937E-04	-.13257E-03	-.25492E+01	.20799E-01	-.15908E+01	-64.5
						.78078E+00	-.33092E+01	.20450E+01	

CONTAINTES DANS L ELEMENT 33

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.15000E+01	.60000E+01	.94548E-17	-.63565E-04	-.19553E-03	-.50852E+00	-.20341E+01	-.23464E+01	-36.0

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						.11959E+01	-.37386E+01	.24672E+01	
2	.15000E+01	.90000E+01	.11538E-02	-.63565E-04	-.32266E-03	.36414E+02	.71965E+01	-.38719E+01	-7.4
						.36918E+02	.66921E+01	.15113E+02	
3	.00000E+00	.90000E+01	.11538E-02	.00000E+00	-.10756E-03	.36922E+02	.92306E+01	-.12907E+01	-2.7
						.36982E+02	.91706E+01	.13906E+02	

CONTAINTES DANS L ELEMENT 34

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.45000E+01	.60000E+01	.10051E-16	.17682E-04	-.48997E-03	.14146E+00	.56584E+00	-.58796E+01	-46.0
						.62371E+01	-.55298E+01	.58834E+01	
2	.45000E+01	.90000E+01	.12376E-02	-.31262E-03	-.29373E-03	.37102E+02	-.10321E+00	-.35248E+01	-5.4
						.37433E+02	-.43421E+00	.18934E+02	
3	.30000E+01	.90000E+01	.12376E-02	-.28854E-03	-.27446E-03	.37294E+02	.66731E+00	-.32935E+01	-5.1
						.37588E+02	.37352E+00	.18607E+02	

CONTAINTES DANS L ELEMENT 35

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.75000E+01	.60000E+01	.10500E-16	-.43041E-06	-.49378E-03	-.34433E-02	-.13773E-01	-.59253E+01	-45.0
						.59167E+01	-.59339E+01	.59253E+01	
2	.75000E+01	.90000E+01	.11511E-02	-.28975E-03	-.27833E-03	.34517E+02	-.63288E-01	-.33400E+01	-5.5
						.34837E+02	-.38293E+00	.17610E+02	
3	.60000E+01	.90000E+01	.11511E-02	-.29304E-03	-.28154E-03	.34491E+02	-.16850E+00	-.33785E+01	-5.5
						.34817E+02	-.49476E+00	.17656E+02	

CONTAINTES DANS L ELEMENT 36

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.10500E+02	.60000E+01	.10764E-16	.24726E-05	-.48140E-03	.19781E-01	.79124E-01	-.57768E+01	-45.1
						.58263E+01	-.57274E+01	.57769E+01	
2	.10500E+02	.90000E+01	.10640E-02	-.26786E-03	-.28220E-03	.31906E+02	-.59492E-01	-.33864E+01	-6.0
						.32260E+02	-.41431E+00	.16337E+02	
3	.90000E+01	.90000E+01	.10640E-02	-.27037E-03	-.28138E-03	.31886E+02	-.13977E+00	-.33766E+01	-6.0
						.32238E+02	-.49191E+00	.16365E+02	

CONTAINTES DANS L ELEMENT 37

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.13500E+02	.60000E+01	.12890E-16	.37887E-05	-.47749E-03	.30310E-01	.12124E+00	-.57299E+01	-45.2
						.58059E+01	-.56543E+01	.57301E+01	
2	.13500E+02	.90000E+01	.97960E-03	-.24677E-03	-.28375E-03	.29373E+02	-.59921E-01	-.34050E+01	-6.5
						.29762E+02	-.44869E+00	.15105E+02	

```
   3  .12000E+02  .90000E+01  .97960E-03 -.24843E-03 -.28119E-03  .29360E+02 -.11298E+00 -.33743E+01      -6.4
                                                                  .29741E+02 -.49436E+00  .15118E+02


CONTAINTES DANS L ELEMENT    38
 P.G.       X          Y          EPSX        EPSY        GAMXY        SIGX        SIGY        TAUXY         TETA
                                                                       SIG1        SIG2        TAUMAX

   1  .16500E+02  .60000E+01  .13957E-16  .41315E-05 -.47656E-03  .33052E-01  .13221E+00 -.57187E+01     -45.2
                                                                  .58015E+01 -.56363E+01  .57189E+01
   2  .16500E+02  .90000E+01  .89602E-03 -.22588E-03 -.28414E-03  .26866E+02 -.60136E-01 -.34097E+01      -7.1
                                                                  .27291E+02 -.48520E+00  .13888E+02
   3  .15000E+02  .90000E+01  .89602E-03 -.22730E-03 -.28114E-03  .26854E+02 -.10537E+00 -.33736E+01      -7.0
                                                                  .27270E+02 -.52112E+00  .13896E+02


CONTAINTES DANS L ELEMENT    39
 P.G.       X          Y          EPSX        EPSY        GAMXY        SIGX        SIGY        TAUXY         TETA
                                                                       SIG1        SIG2        TAUMAX

   1  .19500E+02  .60000E+01  .14105E-16  .42089E-05 -.47635E-03  .33671E-01  .13468E+00 -.57162E+01     -45.3
                                                                  .58006E+01 -.56323E+01  .57164E+01
   2  .19500E+02  .90000E+01  .81263E-03 -.20504E-03 -.28423E-03  .24364E+02 -.60195E-01 -.34107E+01      -7.8
                                                                  .24831E+02 -.52755E+00  .12679E+02
   3  .18000E+02  .90000E+01  .81263E-03 -.20640E-03 -.28112E-03  .24353E+02 -.10360E+00 -.33735E+01      -7.7
                                                                  .24810E+02 -.56039E+00  .12685E+02


CONTAINTES DANS L ELEMENT    40
 P.G.       X          Y          EPSX        EPSY        GAMXY        SIGX        SIGY        TAUXY         TETA
                                                                       SIG1        SIG2        TAUMAX

   1  .22500E+02  .60000E+01  .76941E-17  .42250E-05 -.47631E-03  .33800E-01  .13520E+00 -.57157E+01     -45.3
                                                                  .58004E+01 -.56314E+01  .57159E+01
   2  .22500E+02  .90000E+01  .72929E-03 -.18420E-03 -.28424E-03  .21864E+02 -.60219E-01 -.34109E+01      -8.6
                                                                  .22382E+02 -.57864E+00  .11480E+02
   3  .21000E+02  .90000E+01  .72929E-03 -.18555E-03 -.28112E-03  .21853E+02 -.10322E+00 -.33734E+01      -8.5
                                                                  .22359E+02 -.60984E+00  .11485E+02


CONTAINTES DANS L ELEMENT    41
 P.G.       X          Y          EPSX        EPSY        GAMXY        SIGX        SIGY        TAUXY         TETA
                                                                       SIG1        SIG2        TAUMAX

   1  .25500E+02  .60000E+01  .17713E-16  .42267E-05 -.47629E-03  .33814E-01  .13525E+00 -.57155E+01     -45.3
                                                                  .58002E+01 -.56312E+01  .57157E+01
   2  .25500E+02  .90000E+01  .64595E-03 -.16337E-03 -.28425E-03  .19364E+02 -.60277E-01 -.34109E+01      -9.7
                                                                  .19945E+02 -.64185E+00  .10293E+02
   3  .24000E+02  .90000E+01  .64595E-03 -.16471E-03 -.28112E-03  .19353E+02 -.10316E+00 -.33735E+01      -9.8
                                                                  .19921E+02 -.67148E+00  .10296E+02
```

CONTRAINTES DANS L ELEMENT 42

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.28500E+02	.60000E+01	.12006E-16	.42187E-05	-.47625E-03	.33749E-01	.13500E+00	-.57150E+01	-45.3
						.57996E+01	-.56308E+01	.57152E+01	
2	.28500E+02	.90000E+01	.56263E-03	-.14255E-03	-.28423E-03	.16864E+02	-.60545E-01	-.34108E+01	-11.0
						.17525E+02	-.72206E+00	.91237E+01	
3	.27000E+02	.90000E+01	.56263E-03	-.14388E-03	-.28113E-03	.16853E+02	-.10323E+00	-.33735E+01	-10.8
						.17500E+02	-.74976E+00	.91247E+01	

CONTRAINTES DANS L ELEMENT 43

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.31500E+02	.60000E+01	.17847E-16	.41771E-05	-.47605E-03	.33417E-01	.13367E+00	-.57126E+01	-45.3
						.57964E+01	-.56293E+01	.57129E+01	
2	.31500E+02	.90000E+01	.47934E-03	-.12177E-03	-.28415E-03	.14365E+02	-.61815E-01	-.34098E+01	-12.7
						.15130E+02	-.82716E+00	.79786E+01	
3	.30000E+02	.90000E+01	.47934E-03	-.12308E-03	-.28116E-03	.14354E+02	-.10367E+00	-.33739E+01	-12.5
						.15103E+02	-.85224E+00	.79776E+01	

CONTRAINTES DANS L ELEMENT 44

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.34500E+02	.60000E+01	.12916E-16	.39840E-05	-.47515E-03	.31872E-01	.12749E+00	-.57018E+01	-45.2
						.57817E+01	-.56223E+01	.57020E+01	
2	.34500E+02	.90000E+01	.39623E-03	-.10117E-03	-.28380E-03	.11870E+02	-.67654E-01	-.34056E+01	-14.9
						.12773E+02	-.97085E+00	.68720E+01	
3	.33000E+02	.90000E+01	.39623E-03	-.10236E-03	-.28130E-03	.11860E+02	-.10573E+00	-.33756E+01	-14.7
						.12747E+02	-.99230E+00	.68697E+01	

CONTRAINTES DANS L ELEMENT 45

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.37500E+02	.60000E+01	.74417E-17	.31501E-05	-.47115E-03	.25201E-01	.10080E+00	-.56538E+01	-45.2
						.57169E+01	-.55909E+01	.56539E+01	
2	.37500E+02	.90000E+01	.31391E-03	-.81368E-04	-.28224E-03	.93943E+01	-.92467E-01	-.33868E+01	-17.8
						.10479E+02	-.11775E+01	.58284E+01	
3	.36000E+02	.90000E+01	.31391E-03	-.82072E-04	-.28189E-03	.93886E+01	-.11501E+00	-.33827E+01	-17.7
						.10470E+02	-.11961E+01	.58329E+01	

CONTRAINTES DANS L ELEMENT 46

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.40500E+02	.60000E+01	.59905E-17	.17029E-06	-.45564E-03	.13623E-02	.54493E-02	-.54677E+01	-45.0
						.54711E+01	-.54642E+01	.54677E+01	
2	.40500E+02	.90000E+01	.23489E-03	-.64245E-04	-.27647E-03	.70025E+01	-.17672E+00	-.33177E+01	-21.4
						.83009E+01	-.14751E+01	.48880E+01	
3	.39000E+02	.90000E+01	.23489E-03	-.63479E-04	-.28361E-03	.70086E+01	-.15223E+00	-.34033E+01	-21.8
						.83680E+01	-.15117E+01	.49398E+01	

CONTAINTES DANS L ELEMENT 47

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.43500E+02	.60000E+01	.40635E-16	-.35364E-05	-.42014E-03	-.28291E-01	-.11316E+00	-.50417E+01	-44.8
						.49711E+01	-.51126E+01	.50419E+01	
2	.43500E+02	.90000E+01	.16606E-03	-.48477E-04	-.26673E-03	.49262E+01	-.22277E+00	-.32008E+01	-25.6
						.64594E+01	-.17560E+01	.41077E+01	
3	.42000E+02	.90000E+01	.16606E-03	-.49449E-04	-.28073E-03	.49184E+01	-.25387E+00	-.33688E+01	-26.2
						.65792E+01	-.19147E+01	.42469E+01	

CONTAINTES DANS L ELEMENT 48

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.46500E+02	.60000E+01	.19166E-16	.10056E-03	-.68437E-03	.80452E+00	.32181E+01	-.82125E+01	-49.2
						.10312E+02	-.62894E+01	.83007E+01	
2	.46500E+02	.90000E+01	.85147E-04	.88824E-04	-.41347E-03	.34353E+01	.35235E+01	-.49616E+01	-45.3
						.84413E+01	-.14824E+01	.49618E+01	
3	.45000E+02	.90000E+01	.85147E-04	-.21937E-04	-.13257E-03	.25492E+01	-.20799E-01	-.15908E+01	-25.5
						.33092E+01	-.78078E+00	.20450E+01	

CONTAINTES DANS L ELEMENT 49

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.15000E+01	.90000E+01	.13192E-02	-.18747E-03	-.48780E-03	.40714E+02	.45543E+01	-.58536E+01	-9.0
						.41638E+02	.36303E+01	.19004E+02	
2	.30000E+01	.90000E+01	.11694E-02	-.28854E-03	-.10756E-03	.35114E+02	.12208E+00	-.12907E+01	-2.1
						.35161E+02	.74532E-01	.17543E+02	
3	.15000E+01	.12000E+02	.26425E-02	-.51777E-03	-.44555E-03	.80418E+02	.45714E+01	-.53467E+01	-4.0
						.80793E+02	.41963E+01	.38299E+02	

CONTAINTES DANS L ELEMENT 50

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA
						SIG1	SIG2	TAUMAX	
1	.45000E+01	.90000E+01	.12279E-02	-.30507E-03	-.28349E-03	.36851E+02	.60618E-01	-.34019E+01	-5.2

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX / SIG1	SIGY / SIG2	TAUXY / TAUMAX	TETA
						.37163E+02	-.25130E+00	.18707E+02	
2	.60000E+01	.90000E+01	.11622E-02	-.29304E-03	-.27446E-03	.34847E+02	-.79468E-01	-.32935E+01	-5.3
						.35155E+02	-.38732E+00	.17771E+02	
3	.45000E+01	.12000E+02	.23971E-02	-.59440E-03	-.50558E-04	.71953E+02	.15626E+00	-.60669E+00	-.5
						.71958E+02	.15113E+00	.35903E+02	

CONTAINTES DANS L ELEMENT 51

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.75000E+01	.90000E+01	.11400E-02	-.28810E-03	-.28056E-03	.34177E+02	-.98918E-01	-.33667E+01	-5.6
						.34504E+02	-.42648E+00	.17465E+02	
2	.90000E+01	.90000E+01	.10735E-02	-.27037E-03	-.28154E-03	.32188E+02	-.64231E-01	-.33785E+01	-5.9
						.32538E+02	-.41434E+00	.16476E+02	
3	.75000E+01	.12000E+02	.22432E-02	-.55844E-03	-.50915E-04	.67314E+02	.75302E-01	-.61098E+00	-.5
						.67319E+02	.69751E-01	.33625E+02	

CONTAINTES DANS L ELEMENT 52

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.10500E+02	.90000E+01	.10555E-02	-.26621E-03	-.28314E-03	.31647E+02	-.74499E-01	-.33977E+01	-6.0
						.32006E+02	-.43434E+00	.16220E+02	
2	.12000E+02	.90000E+01	.98726E-03	-.24843E-03	-.28138E-03	.29605E+02	-.51747E-01	-.33766E+01	-6.4
						.29984E+02	-.43134E+00	.15208E+02	
3	.10500E+02	.12000E+02	.20817E-02	-.51677E-03	-.57160E-04	.62479E+02	.11664E+00	-.68592E+00	-.6
						.62486E+02	.10910E+00	.31189E+02	

CONTAINTES DANS L ELEMENT 53

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.13500E+02	.90000E+01	.97190E-03	-.24497E-03	-.28399E-03	.29141E+02	-.63777E-01	-.34078E+01	-6.6
						.29533E+02	-.45616E+00	.14995E+02	
2	.15000E+02	.90000E+01	.90316E-03	-.22730E-03	-.28119E-03	.27083E+02	-.48224E-01	-.33743E+01	-7.0
						.27496E+02	-.46159E+00	.13979E+02	
3	.13500E+02	.12000E+02	.19163E-02	-.47498E-03	-.59055E-04	.57522E+02	.13088E+00	-.70866E+00	-.7
						.57530E+02	.12213E+00	.28704E+02	

CONTAINTES DANS L ELEMENT 54

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.16500E+02	.90000E+01	.88851E-03	-.22403E-03	-.28419E-03	.26640E+02	-.61003E-01	-.34103E+01	-7.2
						.27069E+02	-.48969E+00	.13779E+02	
2	.18000E+02	.90000E+01	.81965E-03	-.20640E-03	-.28114E-03	.24578E+02	-.47407E-01	-.33736E+01	-7.7
						.25032E+02	-.50123E+00	.12766E+02	

3	.16500E+02	.12000E+02	.17499E-02	-.43328E-03	-.59506E-04	.52531E+02	.13437E+00	-.71407E+00	-.8
						.52541E+02	.12464E+00	.26208E+02	

CONTAINTES DANS L ELEMENT 55

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.19500E+02	.90000E+01	.80516E-03	-.20318E-03	-.28424E-03	.24140E+02	-.60378E-01	-.34109E+01	-7.9
						.24611E+02	-.53193E+00	.12572E+02	
2	.21000E+02	.90000E+01	.73628E-03	-.18555E-03	-.28112E-03	.22077E+02	-.47236E-01	-.33735E+01	-8.5
						.22580E+02	-.55019E+00	.11565E+02	
3	.19500E+02	.12000E+02	.15833E-02	-.39161E-03	-.59604E-04	.47533E+02	.13514E+00	-.71525E+00	-.9
						.47544E+02	.12435E+00	.23710E+02	

CONTAINTES DANS L ELEMENT 56

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.22500E+02	.90000E+01	.72183E-03	-.18234E-03	-.28425E-03	.21640E+02	-.60248E-01	-.34110E+01	-8.7
						.22163E+02	-.58378E+00	.11374E+02	
2	.24000E+02	.90000E+01	.65294E-03	-.16471E-03	-.28112E-03	.19577E+02	-.47227E-01	-.33734E+01	-9.5
						.20140E+02	-.61095E+00	.10376E+02	
3	.22500E+02	.12000E+02	.14167E-02	-.34994E-03	-.59628E-04	.42534E+02	.13527E+00	-.71554E+00	-1.0
						.42546E+02	.12320E+00	.21211E+02	

CONTAINTES DANS L ELEMENT 57

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.25500E+02	.90000E+01	.63850E-03	-.16151E-03	-.28424E-03	.19140E+02	-.60238E-01	-.34109E+01	-9.8
						.19728E+02	-.64818E+00	.10188E+02	
2	.27000E+02	.90000E+01	.56961E-03	-.14388E-03	-.28112E-03	.17077E+02	-.47355E-01	-.33735E+01	-10.8
						.17717E+02	-.68797E+00	.92026E+01	
3	.25500E+02	.12000E+02	.12500E-02	-.30827E-03	-.59647E-04	.37534E+02	.13516E+00	-.71576E+00	-1.1
						.37547E+02	.12146E+00	.18713E+02	

CONTAINTES DANS L ELEMENT 58

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.28500E+02	.90000E+01	.55519E-03	-.14068E-03	-.28422E-03	.16641E+02	-.60316E-01	-.34106E+01	-11.1
						.17310E+02	-.72996E+00	.90201E+01	
2	.30000E+02	.90000E+01	.48630E-03	-.12308E-03	-.28113E-03	.14577E+02	-.48011E-01	-.33735E+01	-12.4
						.15318E+02	-.78867E+00	.80531E+01	
3	.28500E+02	.12000E+02	.10833E-02	-.26663E-03	-.59718E-04	.32533E+02	.13443E+00	-.71662E+00	-1.3
						.32549E+02	.11858E+00	.16215E+02	

CONTRAINTES DANS L ELEMENT 59

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX / SIG1	SIGY / SIG2	TAUXY / TAUMAX	TETA
1	.31500E+02	.90000E+01	.47197E-03	-.11989E-03	-.28409E-03	.14144E+02	-.60720E-01	-.34090E+01	-12.8
						.14920E+02	-.83650E+00	.78780E+01	
2	.33000E+02	.90000E+01	.40306E-03	-.10236E-03	-.28116E-03	.12079E+02	-.51108E-01	-.33739E+01	-14.5
						.12954E+02	-.92639E+00	.69403E+01	
3	.31500E+02	.12000E+02	.91654E-03	-.22504E-03	-.60055E-04	.27529E+02	.13093E+00	-.72066E+00	-1.5
						.27548E+02	.11199E+00	.13718E+02	

CONTRAINTES DANS L ELEMENT 60

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX / SIG1	SIGY / SIG2	TAUXY / TAUMAX	TETA
1	.34500E+02	.90000E+01	.38916E-03	-.99246E-04	-.28348E-03	.11659E+02	-.62603E-01	-.34018E+01	-15.1
						.12575E+02	-.97830E+00	.67765E+01	
2	.36000E+02	.90000E+01	.32015E-03	-.82072E-04	-.28130E-03	.95884E+01	-.65076E-01	-.33756E+01	-17.5
						.10652E+02	-.11284E+01	.58900E+01	
3	.34500E+02	.12000E+02	.74941E-03	-.18376E-03	-.61633E-04	.22511E+02	.11486E+00	-.73960E+00	-1.9
						.22535E+02	.90463E-01	.11222E+02	

CONTRAINTES DANS L ELEMENT 61

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX / SIG1	SIGY / SIG2	TAUXY / TAUMAX	TETA
1	.37500E+02	.90000E+01	.30820E-03	-.79265E-04	-.28089E-03	.92283E+01	-.70860E-01	-.33706E+01	-18.0
						.10322E+02	-.11641E+01	.57428E+01	
2	.39000E+02	.90000E+01	.23872E-03	-.63479E-04	-.28189E-03	.71313E+01	-.12156E+00	-.33827E+01	-21.5
						.84640E+01	-.14543E+01	.49592E+01	
3	.37500E+02	.12000E+02	.58057E-03	-.14368E-03	-.68621E-04	.17429E+02	.46786E-01	-.82345E+00	-2.7
						.17468E+02	.78630E-02	.87299E+01	

CONTRAINTES DANS L ELEMENT 62

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX / SIG1	SIGY / SIG2	TAUXY / TAUMAX	TETA
1	.40500E+02	.90000E+01	.23462E-03	-.61830E-04	-.27178E-03	.70132E+01	-.10162E+00	-.32614E+01	-21.3
						.82819E+01	-.13704E+01	.48261E+01	
2	.42000E+02	.90000E+01	.16265E-03	-.49449E-04	-.28361E-03	.48092E+01	-.28117E+00	-.34033E+01	-26.6
						.65138E+01	-.19858E+01	.42498E+01	
3	.40500E+02	.12000E+02	.40429E-03	-.10677E-03	-.95552E-04	.12083E+02	-.18232E+00	-.11466E+01	-5.3
						.12190E+02	-.28859E+00	.62391E+01	

CONTRAINTES DANS L ELEMENT 63

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX	SIGY	TAUXY	TETA

						SIG1	SIG2	TAUMAX	
1	.43500E+02	.90000E+01	.18016E-03	-.49195E-04	-.26112E-03	.53717E+01	-.13292E+00	-.31334E+01	-24.4
						.67899E+01	-.15512E+01	.41705E+01	
2	.45000E+02	.90000E+01	.94911E-04	-.21937E-04	-.28073E-03	.28617E+01	.57312E-01	-.33688E+01	-33.7
						.51084E+01	-.21894E+01	.36489E+01	
3	.43500E+02	.12000E+02	.20152E-03	-.60935E-04	-.15722E-03	.59610E+01	-.33779E+00	-.18866E+01	-15.5
						.64829E+01	-.85964E+00	.36713E+01	

CONTAINTES DANS L ELEMENT 64

P.G.	X	Y	EPSX	EPSY	GAMXY	SIGX SIG1	SIGY SIG2	TAUXY TAUMAX	TETA
1	.46500E+02	.90000E+01	-.11615E-16	.19011E-03	-.25348E-03	.15209E+01	.60834E+01	-.30417E+01	-63.4
						.76043E+01	-.67057E-13	.38021E+01	
2	.48000E+02	.90000E+01	-.73112E-04	.25479E-03	-.13257E-03	-.30126E+00	.75684E+01	-.15908E+01	-79.0
						.78778E+01	-.61068E+00	.42442E+01	
3	.46500E+02	.12000E+02	.73112E-04	-.11770E-03	-.50210E-04	.13979E+01	-.31816E+01	-.60253E+00	-7.4
						.14759E+01	-.32596E+01	.23677E+01	

EQUILIBRIUM RESIDUALS AND REACTIONS

NODES	X	Y	Z	DEGREES OF FREEDOM (* = PRESCRIBED)	
1	.00000E+00	.00000E+00	.00000E+00	.00000E+00 *	.00000E+00 *
2	.00000E+00	.30000E+01	.00000E+00	.00000E+00 *	.00000E+00 *
3	.00000E+00	.60000E+01	.00000E+00	.00000E+00 *	.00000E+00 *
4	.00000E+00	.90000E+01	.00000E+00	.00000E+00 *	.00000E+00 *
5	.00000E+00	.12000E+02	.00000E+00	.00000E+00 *	.00000E+00 *
6	.15000E+01	.00000E+00	.00000E+00	.00000E+00	.00000E+00
7	.15000E+01	.30000E+01	.00000E+00	.00000E+00	.00000E+00
8	.15000E+01	.60000E+01	.00000E+00	.00000E+00	.00000E+00
9	.15000E+01	.90000E+01	.00000E+00	.00000E+00	.00000E+00
10	.15000E+01	.12000E+02	.00000E+00	.00000E+00	.00000E+00
11	.30000E+01	.00000E+00	.00000E+00	.00000E+00	.00000E+00
12	.30000E+01	.30000E+01	.00000E+00	.00000E+00	.00000E+00
13	.30000E+01	.60000E+01	.00000E+00	.00000E+00	.00000E+00
14	.30000E+01	.90000E+01	.00000E+00	.00000E+00	.00000E+00
15	.30000E+01	.12000E+02	.00000E+00	.00000E+00	.00000E+00
16	.45000E+01	.00000E+00	.00000E+00	.00000E+00	.00000E+00
17	.45000E+01	.30000E+01	.00000E+00	.00000E+00	.00000E+00
18	.45000E+01	.60000E+01	.00000E+00	.00000E+00	.00000E+00
19	.45000E+01	.90000E+01	.00000E+00	.00000E+00	.00000E+00
20	.45000E+01	.12000E+02	.00000E+00	.00000E+00	.00000E+00
21	.60000E+01	.00000E+00	.00000E+00	.00000E+00	.00000E+00
22	.60000E+01	.30000E+01	.00000E+00	.00000E+00	.00000E+00
23	.60000E+01	.60000E+01	.00000E+00	.00000E+00	.00000E+00

24	.60000E+01	.90000E+01	.00000E+00	.00000E+00	.00000E+00
25	.60000E+01	.12000E+02	.00000E+00	.00000E+00	.00000E+00
26	.75000E+01	.00000E+00	.00000E+00	.00000E+00	.00000E+00
27	.75000E+01	.30000E+01	.00000E+00	.00000E+00	.00000E+00
28	.75000E+01	.60000E+01	.00000E+00	.00000E+00	.00000E+00
29	.75000E+01	.90000E+01	.00000E+00	.00000E+00	.00000E+00
30	.75000E+01	.12000E+02	.00000E+00	.00000E+00	.00000E+00
31	.90000E+01	.00000E+00	.00000E+00	.00000E+00	.00000E+00
32	.90000E+01	.30000E+01	.00000E+00	.00000E+00	.00000E+00
33	.90000E+01	.60000E+01	.00000E+00	.00000E+00	.00000E+00
34	.90000E+01	.90000E+01	.00000E+00	.00000E+00	.00000E+00
35	.90000E+01	.12000E+02	.00000E+00	.00000E+00	.00000E+00
36	.10500E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
37	.10500E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
38	.10500E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
39	.10500E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
40	.10500E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
41	.12000E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
42	.12000E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
43	.12000E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
44	.12000E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
45	.12000E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
46	.13500E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
47	.13500E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
48	.13500E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
49	.13500E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
50	.13500E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
51	.15000E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
52	.15000E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
53	.15000E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
54	.15000E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
55	.15000E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
56	.16500E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
57	.16500E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
58	.16500E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
59	.16500E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
60	.16500E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
61	.18000E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
62	.18000E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
63	.18000E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
64	.18000E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
65	.18000E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
66	.19500E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
67	.19500E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
68	.19500E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
69	.19500E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
70	.19500E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
71	.21000E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
72	.21000E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
73	.21000E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
74	.21000E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00

75	.21000E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
76	.22500E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
77	.22500E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
78	.22500E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
79	.22500E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
80	.22500E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
81	.24000E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
82	.24000E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
83	.24000E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
84	.24000E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
85	.24000E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
86	.25500E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
87	.25500E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
88	.25500E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
89	.25500E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
90	.25500E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
91	.27000E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
92	.27000E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
93	.27000E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
94	.27000E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
95	.27000E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
96	.28500E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
97	.28500E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
98	.28500E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
99	.28500E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
100	.28500E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
101	.30000E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
102	.30000E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
103	.30000E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
104	.30000E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
105	.30000E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
106	.31500E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
107	.31500E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
108	.31500E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
109	.31500E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
110	.31500E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
111	.33000E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
112	.33000E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
113	.33000E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
114	.33000E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
115	.33000E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
116	.34500E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
117	.34500E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
118	.34500E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
119	.34500E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
120	.34500E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
121	.36000E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
122	.36000E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
123	.36000E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
124	.36000E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
125	.36000E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00

126	.37500E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
127	.37500E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
128	.37500E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
129	.37500E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
130	.37500E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
131	.39000E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
132	.39000E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
133	.39000E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
134	.39000E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
135	.39000E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
136	.40500E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
137	.40500E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
138	.40500E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
139	.40500E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
140	.40500E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
141	.42000E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
142	.42000E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
143	.42000E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
144	.42000E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
145	.42000E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
146	.43500E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
147	.43500E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
148	.43500E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
149	.43500E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
150	.43500E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
151	.45000E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
152	.45000E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
153	.45000E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
154	.45000E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
155	.45000E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
156	.46500E+02	.00000E+00	.00000E+00	.00000E+00	.00000E-00
157	.46500E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
158	.46500E+02	.60000E+01	.00000E+00	.00000E+00	.00000E+00
159	.46500E+02	.90000E+01	.00000E+C0	.00000E+00	.00000E+00
160	.46500E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00
161	.48000E+02	.00000E+00	.00000E+00	.00000E+00	.00000E+00
162	.48000E+02	.30000E+01	.00000E+00	.00000E+00	.00000E+00
163	.48000E+02	.60000E+01	.00000E+00	.00000E+00	.40000E+02
164	.48000E+02	.90000E+01	.00000E+00	.00000E+00	.00000E+00
165	.48000E+02	.12000E+02	.00000E+00	.00000E+00	.00000E+00

END OF PROBLEM,   7001 UTILIZED REAL WORDS OVER   20000

APPENDIX E

MEF PROGRAM LISTINGS

The program listings for MEF are provided below. Each separate disk file, or comoiland, is marked by the Microsoft FORTRAN 77 metacommands which preceed it.

```
$DO66
$NOFLOATCALLS
C
C=====================================================================
C
C      F . E . M .  - 3 -  PROGRAM,  'BOOK'  VERSION   OCTOBER 1979
C     (G.TOUZOT , G.DHATT, COMPIEGNE UNIVERSITY OF TECHNOLOGY, FRANCE)
C
C        MODIFIED FOR IMPLMENTATION ON THE IBM PC AND THE COMUMBIA
C                   MPC USING MICROSOFT FORTRAN VER 3.2
C
C
C      ( REHE E. RUESCH, LCDR, USN, U.S. NAVAL POSTGRADUATE SCHOOL )
C
C=====================================================================
C
C                             MAIN PROGRAM
C
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 BLOC,BLOCS(21)
      CHARACTER*14 INFILE,OUTFILE
      COMMON/COOR/NDIM,NNT,NDLN,NDLT,FAC(3)
      COMMON/COND/NCLT,NCLZ,NCLNZ
      COMMON/PRND/NPRN
      COMMON/PREL/NGPE,NPRE
      COMMON/ELEM/NELT,NNEL,NTPE,NGRE,ME,NIDENT,NPG
      COMMON/ASSE/NSYM,NKG,NKE,NDLE
      COMMON/RESO/NEQ,NRES,MRES
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,
```

```
     1  IPG, ICOD, IDLEO, INELO, IPGO
      COMMON/LIND/NLBL,NBLM,MKG1,MKG2
      COMMON/NLIN/EPSDL,XNORM,OMEGA,XPAS,DPAS,DPAS0,NPAS,IPAS,NITER,
     1  ITER,IMETH
      COMMON/VALP/NITER1,NMDIAG,EPSLB,SHIFT,NSS,NSWM,TOLJAC,NVALP
      COMMON/ES/M,MR,MP,MLUN(10)
      COMMON/ALLOC/NVA,IVA,IVAMAX,NREEL,NTBL
      COMMON/LOC/LCORG,LDLNC,LNEQ,LDIMP,LPRNG,LPREG,LLD,LLOCE,LCORE,LNE,
     1  LPRNE,LPREE,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,LRES,LDLG,LME,
     2  LDLEO,LDLGO,LFGO
      COMMON/TRVL/VDE(9),RDUMMY(512),NULL
      COMMON/DUMPLA/Y13,Y21,X13,X21,SU4,SU5,SU6,D4,D5,D6,
     1CL4,CL5,CL6,SL4,SL5,SL6,B(3,9)
      COMMON VA(20000)
      DATA BLOCS/'IMAG','COMT','COOR','DLPN','COND','PRND','PREL',
     *           'ELEM','SOLC','SOLR','LINM','LIND','NLIN','TEMP',
     *           'VALP','....','....','....','....','....','STOP'/
      DATA NB/21/
C
C+++++++++  WRITE HEADING TO CONSOLE AND REQUEST INPUT AND OUTPUT
C           FILE NAMES.  FILE NAMES MUST CONFORM TO MS DOS 2.0
C           CONVENTIONS.  NO PATHNAMES ALLOWED.  A NAME CAN, THEREFORE,
C           CONSIST OF (AT MOST) 14 CHARACTERS: DEV:FILENAME.EXT
C
C                      FOR EXAMPLE:   A:INPUT.DAT
C
      WRITE(*,2000)
      WRITE(*,'(A\)')   ' COMMAND-FILE NAME? '
      READ(*,'(A14)') INFILE
      WRITE(*,'(/)')
      WRITE(*,'(A\)')   ' OUTPUT FILE NAME?  '
      READ(*,'(A14)') OUTFILE
      WRITE(*,'(/)')
      WRITE(*,'(A)')    ' PROCESSING BEGINS...'
      OPEN(MP,FILE=OUTFILE,STATUS='NEW')
      OPEN(MR,FILE=INFILE,STATUS='OLD')
C
C++++++++
C
C-----------------------------------------------------------------------
C....... LENGTH OF BLANK COMMON IN REAL WORDS (TABLE VA)
      NVA=20000
C------- HEADING
      WRITE(MP,2000)
2000  FORMAT(1H1,30X,'F.E.M.3.'/25X,' G.TOUZOT, G.DHATT'
     1                       ,/25X,'    MODIFIED BY'
     2                      ,//25X,'   RENE E. RUESCH'/25X,19('=')//)
C------ READ BLOCK TITLE
10    READ(MR,1000) BLOC,M,MLUN
      WRITE(*,'(A18,A4)') ' PROCESSING BLOCK ', BLOC
1000  FORMAT(A4,I6,10I5)
```

```
C------ SEARCH FOR THE BLOCK TO BE EXECUTED
      DO 20 I=1,NB
        IF(BLOC.EQ.BLOCS(I)) GO TO 30
20    CONTINUE
      WRITE(MP,2010)
2010  FORMAT(' ** ERROR, MISSING BLOCK CALLING CARD',/)
      GO TO 10
30    GO TO (110,120,130,140,150,160,170,
     1       180,190,200,210,220,230,240,
     2       250,260,270,280,290,300,999),I
C------ BLOCK TO PRINT IMAGES OF DATA CARDS               'IMAG'
110   CALL BLIMAG
      GO TO 10
C------ BLOCK TO READ AND PRINT COMMENTS                  'COMT'
120   CALL BLCOMT
      GO TO 10
C------ BLOCK TO READ NODAL POINTS COORDINATES            'COOR'
130   CALL BLCOOR
      GO TO 10
C------ BLOCK TO READ DEGREES OF FREEDOM PER NODE         'DLPN'
140   CALL BLDLPN
      GO TO 10
C------ BLOCK TO READ BOUNDARY CONDITIONS                 'COND'
150   CALL BLCOND
      GO TO 10
C------ BLOCK TO READ NODAL PROPERTIES                    'PRND'
160   CALL BLPRND
      GO TO 10
C------ BLOCK TO READ ELEMENT PROPERTIES                  'PREL'
170   CALL BLPREL
      GO TO 10
C------ BLOCK TO READ ELEMENT DATA                        'ELEM'
180   CALL BLELEM
      GO TO 10
C------ BLOCK TO READ CONCENTRATED LOADS                  'SOLC'
190   CALL BLSOLC
      GO TO 10
C------ BLOCK TO READ DISTRIBUTED LOADS                   'SOLR'
200   CALL BLSOLR
      GO TO 10
C------ BLOCK FOR IN CORE ASSEMBLING AND LINEAR SOLUTION 'LINM'
210   CALL BLLINM
      GO TO 10
C------ BLOCK FOR ON DISK ASSEMBLING AND LINEAR SOLUTION 'LIND'
220   CALL BLLIND
      GO TO 10
C------ BLOCK FOR NON LINEAR PROBLEM SOLUTION             'NLIN'
230   CALL BLNLIN
      GO TO 10
C------ BLOCK FOR UNSTEADY PROBLEM                        'TEMP'
240   CALL BLTEMP
```

```
      GO TO 10
C------- BLOCK TO COMPUTE EIGENVALUES (SUBSPACE)          'VALP'
250   CALL BLVALP
      GO TO 10
C------- UNDEFINED BLOCS
250   CONTINUE
270   CONTINUE
280   CONTINUE
290   CONTINUE
300   CONTINUE
      GO TO 10
C------- END OF PROBLEM                                   'STOP'
999   WRITE(MP,2020) IVAMAX,NVA
2020  FORMAT(//' END OF PROBLEM, ',I10,' UTILIZED REAL WORDS OVER',I10)
      STOP
      END
```

```
$LARGE
$NOFLOATCALLS
      BLOCK DATA
C=================================================================
C     INITIALIZE LABELLED COMMONS
C=================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNT,NDLN,NDLT,FAC(3)
      COMMON/COND/NCLT,NCLZ,NCLNZ
      COMMON/PRND/NPRN
      COMMON/PREL/NGPE,NPRE
      COMMON/ELEM/NELT,NNEL,NTPE,NGRE,ME,NIDENT,NPG
      COMMON/ASSE/NSYM,NKG,NKE,NDLE
      COMMON/RESO/NEQ,NRES,MRES
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,
     1  IPG,ICOD,IDLEO,INELO,IPGO
      COMMON/LIND/NLBL,NBLM,MKG1,MKG2
      COMMON/NLIN/EPSDL,XNORM,OMEGA,XPAS,DPAS,DPASO,NPAS,IPAS,NITER,
     1  ITER,IMETH
      COMMON/VALP/NITER1,NMDIAG,EPSLB,SHIFT,NSS,NSWM,TOLJAC,NVALP
      COMMON/ES/M,MR,MP,MLUN(10)
      COMMON/ALLOC/NVA,IVA,IVAMAX,NREEL,NTBL
      COMMON/LOC/LCORG,LDLNC,LNEQ,LDIMP,LPRNG,LPREG,LLD,LLOCE,LCORE,LNE,
     1  LPRNE,LPREE,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,LRES,LDLG,LME,
     2  LDLEO,LDLGO,LFGO
      DIMENSION LCORG(1),LDLNC(1),LNEQ(1),LDIMP(1),LPRNG(1),LPREG(1),
     *  LLD(1),LLOCE(1),LCORE(1),LNE(1),LPRNE(1),LPREE(1),LDLE(1),
     *  LKE(1),LFE(1),LKGS(1),LKGD(1),LKGI(1),LFG(1),LRES(1),LDLG(1),
     *  LME(1),LDLEO(1),LDLGO(1),LFGO(1)
      DIMENSION LXX(25)
      EQUIVALENCE (LXX(1),LCORG)
C-------  COMMON /COOR/
      DATA NNT/20/,NDLN/2/,NDIM/2/,FAC(1),FAC(2),FAC(3)/3*1.D0/
C-------  COMMON /PRND/
      DATA NPRN/0/
C-------  COMMON /PREL/
      DATA NGPE/0/,NPRE/0/
C-------  COMMON /ELEM/
      DATA NELT/20/,NNEL/8/,NTPE/1/,NGRE/1/,ME/1/,NIDENT/0/
C-------  COMMON/ASSE/
      DATA NSYM/0/
C-------  COMMON /RESO/
      DATA NRES/0/,MRES/2/
C-------  COMMON /RGDT/
      DATA ITPE1/0/
C-------  COMMON /LIND/
      DATA MKG1/4/,MKG2/7/
C-------  COMMON /NLIN/
      DATA EPSDL/1.D-2/,OMEGA/1.D0/,DPAS/.2D0/,NPAS/1/,NITER/5/,IMETH/1/
C-------  COMMON /VALP/
```

```
      DATA NITER1/10/,NMDIAG/0/,EPSLB/1.D-3/,SHIFT/0.D0/,NSS/5/,
     1  NSWM/12/,TOLJAC/1.D-12/,NVALP/3/
C------  COMMON /ES/
      DATA MR/5/,MP/6/
C------  COMMON /ALLOC/
      DATA IVA/1/,IVAMAX/1/,NTBL/25/
C.......  DEFINE HERE THE NUMBER OF INTEGERS CONTAINED IN A REAL
C         FOR THE COMPUTER EMPLOYED
C           EXAMPLES:  IBM   SIMPLE PRECISION    NREEL.EQ.1
C                      IBM   DOUBLE PRECISION    NREEL.EQ.2
C                      CDC                       NREEL.EQ.1
      DATA NREEL/2/
C.......
C------  COMMON /LOC/
      DATA LXX(1),LXX(2),LXX(3),LXX(4),LXX(5),LXX(6),LXX(7),LXX(8),
     1     LXX(9),LXX(10),LXX(11),LXX(12),LXX(13),LXX(14),LXX(15),
     2     LXX(16),LXX(17),LXX(18),LXX(19),LXX(20),LXX(21),LXX(22),
     3     LXX(23),LXX(24),LXX(25)/25*1/
      END
```

```
$LARGE
$NOFLOATCALLS
$DEBUG
      SUBROUTINE ERREUR(IERR,I1,I2,INIV)
C======================================================================
C     PRINT ERROR MESSAGES FOR BLOCKS READING DATA
C======================================================================
      COMMON/ES/M,MR,MP,MDUMMY(10)
C----------------------------------------------------------------------
C------- BLOCK 'COOR'
      IF(IERR.GT.19) GO TO 200
      IE=IERR-10
      GO TO (110,120,130,140,150,160,160,180),IE
110   WRITE(MP,2110)I1,I2
2110  FORMAT(' *** ERROR, FIRST NODE NUMBER(',I4,') IS GREATER THAN NNT=
     1',I4)
      GO TO 900
120   WRITE(MP,2120)I1,I2
2120  FORMAT(' ** ERROR, SECOND NODE NUMBER(',I4,') IS GREATER THAN NNT=
     1',I4)
      GO TO 900
130   WRITE(MP,2130)I1,I2
2130  FORMAT(' ** ERROR, NODAL NUMBER OF D.O.F.(',I4,') IS GREATER THAN
     1NDLN=',I4)
      GO TO 900
140   WRITE(MP,2140)
2140  FORMAT(' ** ERROR, FIRST AND SECOND NODE NUMBERS ARE INCOMPATIBLE
     1WITH THE GENERATION PARAMETER')
      GO TO 900
150   WRITE(MP,2150)I1
2150  FORMAT(' ** ERROR, NODE ',I4,' IS DEFINED MORE THAN ONCE')
      GO TO 900
160   WRITE(MP,2160)I1
2160  FORMAT(' ** ERROR, NODE ',I4,' IS NOT DEFINED')
      GO TO 900
180   WRITE(MP,2180)I2,I1
2180  FORMAT(' ** ERROR, GENERATED NODES NUMBER(',I4,') IS LESS THAN NNT
     1=',I4)
      GO TO 900
C------- BLOCK 'DLPN'
200   IF(IERR.GT.29) GO TO 300
      IE=IERR-20
      GO TO (210,220),IE
210   WRITE(MP,2210)I1,I2
2210  FORMAT(' ** ERROR, NUMBER OF D.O.F. (',I2,') IS GREATER THAN NDLN=
     1',I2)
      GO TO 900
220   WRITE(MP,2220)I1,I2
2220  FORMAT(' ** ERROR, NODE NUMBER(',I4,') IS GREATER THAN
     1NNT=',I4)
```

```
      GO TO 900
C------- BLOCK 'COND'
300   IF(IERR.GT.39)GO TO 400
      IE=IERR-30
      GO TO (900,320,900),IE
320   GO TO 220
C------- BLOCK 'PREL'
400   IF(IERR.GT.49) GO TO 500
      IE=IERR-40
      GO TO (410,900),IE
410   WRITE(MP,2410)I1,I2
2410  FORMAT(' ** ERROR, GROUP NUMBER (',I3,') IS GREATER THAN NGPE=',I3
     1)
      GO TO 900
C------- BLOCK 'ELEM'
500   IF(IERR.GT.59) GO TO 900
      IE=IERR-50
      GO TO (510,900,530,540,550,560,570),IE
510   WRITE(MP,2510)I1,I2
2510  FORMAT(' ** ERROR, NUMBER OF NODES (',I3,') IS GREATER THAN NNEL='
     1,I3)
      GO TO 900
530   WRITE(MP,2530)I1,I2
2530  FORMAT(' ** ERROR, PROPERTY NUMBER (',I3,') IS GREATER THAN NGPE='
     ,1I3)
      GO TO 900
540   WRITE(MP,2540)I1,I2
2540  FORMAT(' ** ERROR, GROUP NUMBER (',I3,') IS GREATER THAN NGRE=',I3
     1)
      GO TO 900
550   WRITE(MP,2550)I1,I2
2550  FORMAT(' ** ERROR, ELEMENT NUMBER (',I4,') IS GREATER THAN NELT=',
     1I4)
      GO TO 900
560   GO TO 220
570   WRITE(MP,2570)I1,I2
2570  FORMAT(' ** ERROR, NUMBER OF ELEMENTS (',I4,') IS GREATER THAN NEL
     1T=',I4)
C------- END
900   I1=I2
      IF(INIV.GE.2) STOP
      RETURN
      END
```

```
      SUBROUTINE ESPACE(ILONG,IREEL,TBL,IDEB)
C=====================================================================
C     TO ALLOCATE A REAL OR INTEGER TABLE IN ARRAY VA
C      INPUT
C         ILONG             LENGTH OF THE TABLE TO BE ALLOCATED
C                           (IN REAL OR INTEGER WORDS)
C         IREEL             TABLE TYPE :
C                                .EQ.0     INTEGER
C                                .EQ.1     REAL
C         TBL               NAME OF THE TABLE (A4)
C      OUTPUT
C         IDEB              TABLE TO BE ALLOCATED STARTS IN VA(IDEB)
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/ES/M,MR,MP,MDUMMY(10)
      COMMON/ALLOC/NVA,IVA,IVAMAX,NREEL,IDUMMY
      COMMON VA(1)
      DIMENSION KA(1)
      EQUIVALENCE (VA(1),KA(1))
      DATA ZERO/0.D0/
C---------------------------------------------------------------------
C------- CALCULATE THE TABLE LENGTH IN REAL WORDS
      ILGR=ILONG
      IF(IREEL.EQ.0) ILGR=(ILONG+NREEL-1)/NREEL
      IVA1=IVA+ILGR
C------- CHECK IF ENOUGH SPACE IS AVAILABLE
      IF(IVA1.LE.NVA) GO TO 20
C....... AUTOMATIC EXTENSION OF THE BLANK COMMON IF CORRESPONDING
C        SYSTEM COMMAND EXIST ON THE COMPUTER USED
C     CALL EXTEND(IVA1,IERR)
C     IF(IERR.EQ.1) GO TO 10
C     NVA=IVA1
C     GO TO 20
C------- ALLOCATION ERROR (NOT ENOUGH SPACE)
10    WRITE(MP,2000) TBL,IVA1,NVA
2000  FORMAT(' **** ALLOCATION ERROR, TABLE ',A4/' REQUIRED SPACE:',I9,'
     1 REAL WORDS, AVAILABLE SPACE:',I9,' REAL WORDS')
      STOP
C------- ALLOCATE TABLE
20    IDEB=IVA+1
      IVA=IVA1
      IF(IVA.GT.IVAMAX) IVAMAX=IVA
      IF(M.GT.0) WRITE(MP,2010) TBL,IDEB,IVA1
2010  FORMAT(60X,'TABLE ',A4,' GOES FROM VA(',I7,') TO VA(',I7,')')
C------- INITIALIZE THE ALLOCATED TABLE TO ZERO
      I1=IDEB
      IF(IREEL.EQ.0) I1=(I1-1)*NREEL+1
      I2=I1+ILONG-1
      IF(IREEL.EQ.0) GO TO 40
```

```
      DO 30 I=I1,I2
30    VA(I)=ZERO
      RETURN
40    DO 50 I=I1,I2
50    KA(I)=0
      RETURN
      END


      SUBROUTINE VIDE(IDEB,IREEL,TBL)
C======================================================================
C     TO DELETE A TABLE FROM VA, FOLLOWED BY COMPACTING
C       INPUT
C          IDEB               FIRST POSITION OF TABLE TO BE DELETED
C          IREEL              TYPE OF TABLE (SEE ESPACE)
C          TBL                NAME OF THE TABLE (A4)
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/ES/M,MR,MP,MDUMMY(10)
      COMMON/ALLOC/NVA,IVA,IVAMAX,NREEL,NTBL
      COMMON/LOC/LXX(25)
      COMMON VA(1)
C----------------------------------------------------------------------
C------- SEARCH FOR THE FIRST POSITION OF NEXT TABLE
      I1=IVA+1
      DO 10 I=1,NTBL
      IF(LXX(I).LE.IDEB) GO TO 10
      IF(LXX(I).LT.I1) I1=LXX(I)
10    CONTINUE
C------- SHIFT ALL TABLES AFTER THIS
      ID=I1-IDEB
      IF(I1.EQ.IVA+1) GO TO 40
      DO 20 I=1,NTBL
      IF(LXX(I).GT.IDEB) LXX(I)=LXX(I)-ID
20    CONTINUE
      DO 30 I=I1,IVA
      J=I-ID
30    VA(J)=VA(I)
C------- PRINT
40    IVA=IVA-ID
      IF(M.GT.0) WRITE(MP,2000) TBL,ID,IDEB
2000  FORMAT(60X,'DELETED TABLE ',A4,' COMPACTING ',I7,' REAL WORDS AFTE
     1R VA(',I7,')')
      RETURN
      END
```

```
      SUBROUTINE BLIMAG
C=======================================================================
C     TO CALL AND EXECUTE BLOCK 'IMAG'
C     TO PRINT OUT THE IMAGE OF DATA CARDS
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ES/M,MR,MP,M1,MDUMMY(9)
      COMMON/TRVL/CART(20),RDUMMY(501),NULL
      DATA ICARTM/40/
C-----------------------------------------------------------------------
      IF(M1.EQ.0) M1=MR
      WRITE(MP,2000)
2000  FORMAT(///,1X,'IMAGE OF DATA CARDS'/1X,28('='),/)
      WRITE(MP,2005)
2005  FORMAT(/
     1 50X,'C O L U M N   N U M B E R',/,13X,'CARD',8X,
     2 10X,'1',9X,'2',9X,'3',9X,'4',9X,'5',9X,'6',9X,'7',9X,'8',/,
     3 12X,'NUMBER',8X,8('1234567890'),/,12X,8('-'),6X,80('-'))
      ICART=0
      ICART1=0
10    READ(M1,1000,END=30) CART
1000  FORMAT(20A4)
      ICART=ICART+1
      ICART1=ICART1+1
      IF(ICART1.LE.ICARTM) GO TO 20
      WRITE(MP,2010)
2010  FORMAT(12X,8(1H-),6X,80(1H-),/,13X,'CARD',9X,8('1234567890'),/,
     1 12X,'NUMBER',8X,9X,'1',9X,'2',9X,'3',9X,'4',9X,'5',9X,'6',
     2 9X,'7',9X,'8',/,50X,'C O L U M N   N U M B E R')
      WRITE(MP,2015)
2015  FORMAT(1H1,//)
      WRITE(MP,2005)
      ICART1=0
20    WRITE(MP,2020) ICART,CART
2020  FORMAT(10X,I10,6X,20A4)
      GO TO 10
30    WRITE(MP,2010)
      WRITE(MP,2030)
2030  FORMAT(///,51X,'E N D   O F   D A T A',/,1H1)
      REWIND M1
      READ(M1,1000) CART
      RETURN
      END
```

```
      SUBROUTINE BLCOMT
C=====================================================================
C     TO CALL AND EXECUTE BLOCK 'COMT'
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 BLANC,CART
      COMMON/ES/M,MR,MP,MDUMMY(10)
      COMMON/TRVL/CART(20),RDUMMY(511),NULL
      DATA BLANC/'    '/
C---------------------------------------------------------------------
      WRITE(MP,2000)
2000  FORMAT(//' COMMENTS'/' ',10('=')/)
C------- READ A COMMENT CARD
10    READ(MR,1000) CART
1000  FORMAT(20A4)
C------- SEARCH FOR A WHOLLY BLANK CARD
      DO 20 I=1,20
      IF(CART(I).NE.BLANC) GO TO 30
20    CONTINUE
      RETURN
30    WRITE(MP,2010) CART
2010  FORMAT(1X,20A4)
      GO TO 10
      END
```

```
      SUBROUTINE BLCOOR
C======================================================================
C     TO CALL BLOCK COOR
C     TO READ NODAL COORDINATES
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/COOR/NDIM,NNT,NDLN,NDLT,FAC(3)
      COMMON/ES/M,MR,MP,M1,MDUMMY(9)
      COMMON/ALLOC/NVA,IDUMMY(4)
      COMMON/LOC/LCORG,LDLNC,LDUMMY(23)
      COMMON/TRVL/FAC1(3),IN(3),RDUMMY(517)
      COMMON VA(1)
      DIMENSION TBL(2)
      DATA ZERO/0.D0/
C
C+++     THIS IS COMMENTED OUT BECAUSE OF THE MS FORTRAN COMP-
C+++     ILER BUG SHICH WILL NOT INITIALIZE $LARGE ARRAYS
C+++     THIS ARRAY IS NOW INITIALIZED BY A CALL TO INITBL WHICH
C+++     EXISTS SOLELY TO INITIALIZE TABLE NAMES.
C
C     DATA TBL/'CORG','DLNC'/
C
C        HERE IS THE CALL TO GET AROUND THE COMPILER BUG
C
      CALL INITBL(TBL,'COOR')
C
C+++     ALL OF THIS WAS TO GET AROUND THE MICROSOFT
C+++     COMPILER BUG
C
C----------------------------------------------------------------------
C-------  BLOCK HEADING
      IF(M1.EQ.0) M1=MR
      READ(M1,1000) IN,FAC1
1000  FORMAT(3I5,3F10.0)
C-------  DEFAULT OPTIONS
      IF(IN(1).GT.0) NNT=IN(1)
      IF(IN(2).GT.0) NDLN=IN(2)
      IF(IN(3).GT.0) NDIM=IN(3)
      DO 10 I=1,3
      IF(FAC1(I).NE.ZERO) FAC(I)=FAC1(I)
10    CONTINUE
C-------  PRINT BLOCK PARAMETERS
      WRITE(MP,2000) M,NNT,NDLN,NDIM,FAC,NVA
2000  FORMAT(//' INPUT OF NODES (M=',I2,')'/' ',18('=')/
     1  15X,'MAX. NUMBER OF NODES                (NNT)=',I5/
     2  15X,'MAX. NUMBER OF D.O.F. PER NODE     (NDLN)=',I5/
     3  15X,'DIMENSIONS OF THE PROBLEM          (NDIM)=',I5/
     4  15X,'COORDINATE SCALE FACTORS            (FAC)=',3E12.5/
     5  15X,'WORKSPACE IN REAL WORDS             (NVA)=',I10)
```

```
C------- ALLOCATE SPACE
      IF(LCORG.EQ.1) CALL ESPACE(NNT*NDIM,1,TBL(1),LCORG)
      IF(LDLNC.EQ.1) CALL ESPACE(NNT+1,0,TBL(2),LDLNC)
C------- EXECUTE THE BLOCK
      CALL EXCOOR(VA(LCORG),VA(LDLNC))
      RETURN
      END


      SUBROUTINE EXCOOR(VCORG,KDLNC)
C=====================================================================
C     TO EXECUTE BLOCK 'COOR'
C     READ NODAL COORDINATES
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNT,NDLN,NDLT,FAC(3)
      COMMON/ES/M,MR,MP,M1,MDUMMY(9)
      COMMON/TRVL/X1(3),X2(3),RDUMMY(515),NULL
      DIMENSION VCORG(*),KDLNC(*)
      DATA SPECL/1.23456789D31/
C---------------------------------------------------------------------
C------- INITIALIZE COORDINATES
      I1=(NNT-1)*NDIM+1
      DO 10 I=1,I1,NDIM
10    VCORG(I)=SPECL
C------- READ NODAL DATA CARDS
      IF(M.GT.0) WRITE(MP,2000)
2000  FORMAT(//' NODAL DATA CARDS'/)
20    READ(M1,1000) IN1,X1,IN2,X2,INCR,IDLN
1000  FORMAT(2(I5,3F10.0),2I5)
      IF(M.GT.0) WRITE(MP,2010) IN1,X1,IN2,X2,INCR,IDLN
2010  FORMAT(' >>>>>',2(I5,3E12.5),2I5)
      IF(IN1.LE.0) GO TO 60
C------- DECODE THE CARD
      IF(IN1.GT.NNT) CALL ERREUR(11,IN1,NNT,0)
      IF(IN2.GT.NNT) CALL ERREUR(12,IN2,NNT,0)
      IF(IN2.LE.0) IN2=IN1
      IF(IDLN.GT.NDLN) CALL ERREUR(13,IDLN,NDLN,0)
      IF(IDLN.LE.0) IDLN=NDLN
      IF(INCR.EQ.0) INCR=1
      I1=(IN2-IN1)/INCR
      I2=IN1+I1*INCR
      IF(I1.EQ.0)I1=1
      IF(IN2.NE.I2) CALL ERREUR(14,IN2,IN2,0)
C------- GENERATE NODES BY INTERPOLATION
      DO 30 I=1,NDIM
      X1(I)=X1(I)*FAC(I)
      X2(I)=X2(I)*FAC(I)
30    X2(I)=(X2(I)-X1(I))/I1
      I1=0
      I2=(IN1-1)*NDIM+1
```

```
      I3=(INCR-1)*NDIM
      DO 50 IN=IN1,IN2,INCR
      KDLNC(IN+1)=IDLN
      IF(VCORG(I2).NE.SPECL) CALL ERREUR(15,IN,IN,0)
      DO 40 I=1,NDIM
      VCORG(I2)=X1(I)+X2(I)*I1
40    I2=I2+1
      I1=I1+1
50    I2=I2+I3
      GO TO 20
C------ CHECK FOR MISSING NODES
60    I1=NNT*NDIM+1
      I2=0
      I3=NNT+1
      DO 90 I=1,NNT
      I1=I1-NDIM
      I3=I3-1
      IF(VCORG(I1)-SPECL) 70,80,70
70    IF(I2.EQ.0) I2=I3
      GO TO 90
80    IF(I2.EQ.0) CALL ERREUR(16,I3,I3,0)
      IF(I2.NE.0) CALL ERREUR(17,I3,I3,1)
90    CONTINUE
      IF(I2.NE.NNT) CALL ERREUR(18,NNT,I2,0)
C------- TOTAL NUMBER OF D.O.F.
      NDLT=0
      I1=NNT+1
      DO 100 I=2,I1
100   NDLT=NDLT+KDLNC(I)
C------- OUTPUT
      IF(M.LT.2) GO TO 120
      WRITE(MP,2020)
2020  FORMAT(/10X,'NODE D.O.F.',5X,'X',11X,'Y',11X,'Z'/)
      I1=1
      I2=NDIM
      DO 110 IN=1,NNT
      WRITE(MP,2030) IN,KDLNC(IN+1),(VCORG(I),I=I1,I2)
2030  FORMAT(10X,2I5,3E12.5)
      I1=I1+NDIM
110   I2=I2+NDIM
120   RETURN
      END
```

```
      SUBROUTINE BLDLPN
C=======================================================================
C     TO CALL BLOCK 'DLPN'
C     TO READ NUMBER OF D.O.F. PER NODE
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ES/M,MR,MP,M1,MDUMMY(9)
      COMMON/LOC/LCORG,LDLNC,LDUMMY(23)
      COMMON VA(1)
C-----------------------------------------------------------------------
      IF(M1.EQ.0) M1=MR
      WRITE(MP,2000) M
2000  FORMAT(//' INPUT OF D.O.F. (M=',I2,')'/' ',17('='))
      CALL EXDLPN(VA(LDLNC))
      RETURN
      END
```

```
      SUBROUTINE EXDLPN(KDLNC)
C======================================================================
C     TO EXECUTE BLOCK 'DLPN'
C     TO READ THE NUMBER OF D.O.F. PER NODE
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNT,NDLN,NDLT,FNULL(3)
      COMMON/ES/M,MR,MP,M1,MDUMMY(9)
      COMMON/TRVL/K1(15),RDUMMY(514)
      DIMENSION KDLNC(*)
C----------------------------------------------------------------------
      IF(M.GT.0) WRITE(MP,2000)
2000  FORMAT(//'GROUP OF D.O.F.'/)
C------- READ A GROUP CARD
10    READ(M1,1000) IDLN,K1
1000  FORMAT(16I5)
      IF(M.GT.0) WRITE(MP,2010)IDLN,K1
2010  FORMAT(' >>>>>',16I5)
      IF(IDLN.LE.0) GO TO 40
      IF(IDLN.GT.NDLN) CALL ERREUR(21,IDLN,NDLN,1)
C------- STORE D.O.F. NUMBERS
20    DO 30 I=1,15
      J=K1(I)
      IF(J.LE.0) GO TO 10
      IF(J.GT.NNT) CALL ERREUR(22,J,NNT,1)
30    KDLNC(J+1)=IDLN
      READ(M1,1010) K1
1010  FORMAT(5X,15I5)
      IF(M.GT.0) WRITE(MP,2020) K1
2020  FORMAT(' >>>>>',5X,15I5)
      GO TO 20
C------- TOTAL NUMBER OF D.O.F.
40    NDLT=0
      J=NNT+1
      DO 50 I=2,J
50    NDLT=NDLT+KDLNC(I)
      RETURN
      END
```

```
      SUBROUTINE BLCOND
C=========================================================================
C     TO CALL BLOCK 'COND'
C     TO READ BOUNDARY CONDITIONS AND GENERATE TABLE (NEQ)
C=========================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/COOR/NDIM,NNT,NDLN,NDLT,FNULL(3)
      COMMON/COND/NCLT,NCLZ,NCLNZ
      COMMON/ALLOC/NVA,IVA,IDUMMY(3)
      COMMON/ES/M,MR,MP,M1,MDUMMY(9)
      COMMON/LOC/LCORG,LDLNC,LNEQ,LDIMP,LDUMMY(21)
      COMMON VA(1)
      DIMENSION TBL(2)
C
C+++     THIS IS COMMENTED OUT BECAUSE OF THE MS FORTRAN COMP-
C+++     ILER BUG WHICH WILL NOT INITIALIZE *LARGE ARRAYS
C+++     THIS ARRAY IS NOW INITIALIZED BY A CALL TO INITBL WHICH
C+++     EXISTS SOLELY TO INITIALIZE TABLE NAMES.
C
C     DATA TBL/'NEQ ','DIMP'/
C
C        HERE IS THE CALL TO GET AROUND THE COMPILER BUG
C
      CALL INITBL(TBL,'COND')
C
C+++     ALL THIS WAS SIMPLY TO GET AROUND THE MICROSOFT
C+++     COMPILER BUG
C
C-------------------------------------------------------------------------
      IF(M1.EQ.0) M1=MR
      WRITE(MP,2000) M
2000  FORMAT(//' INPUT OF BOUNDARY CONDITIONS (M=',I2,')'/' ',
     1  33('=')/)
      IF(LNEQ.EQ.1) CALL ESPACE(NDLT,0,TBL(1),LNEQ)
      IF(LDIMP.EQ.1) CALL ESPACE(NDLT,1,TBL(2),LDIMP)
      CALL EXCOND(VA(LCORG),VA(LDLNC),VA(LNEQ),VA(LDIMP))
      CALL VIDE(LDIMP+NCLT,1,TBL(2))
      RETURN
      END
```

```
      SUBROUTINE EXCOND(VCORG,KDLNC,KNEQ,VDIMP)
C======================================================================
C     TO EXECUTE BLOCK 'COND'
C     READ BOUNDARY CONDITIONS AND GENERALE TABLE (NEQ)
C======================================================================
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/COOR/NDIM,NNT,NDLN,NDLT,FNULL(3)
      COMMON/COND/NCLT,NCLZ,NCLNZ
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/ES/M,MR,MP,M1,MDUMMY(9)
      COMMON/TRVL/ KV(16),V(10),H(20),ICOD(10),RDUMMY(478),NULL
      DIMENSION VCORG(*),KDLNC(*),KNEQ(*),VDIMP(*)
      DATA L7/7/,L8/8/,L16/16/ ,X1/0.0D0/,X2/0.0D0/,X3/0.0D0/,ZERO/0.D0/
C----------------------------------------------------------------------
C------- CUMULATIVE TABLE KDLNC
      DO 10 IN=1,NNT
10    KDLNC(IN+1)=KDLNC(IN)+KDLNC(IN+1)
      I1=NNT+1
      IF(M.GE.2) WRITE(MP,2000) (KDLNC(IN),IN=1,I1)
2000  FORMAT(//' NUMBER OF D.O.F. PRECEDING EACH NODE   (DLNC)'/
     1  (1X,10I10))
C------- INITIALIZE
      NCLT=0
      NCLNZ=0
      NCLZ=0
      IF(M.GE.0) WRITE(MP,2010)
2010  FORMAT(//' BOUNDARY CONDITIONS CARDS'/)
C------- READ A B.C. GROUP CARD : 10 CODES + PRESCRIBED VAL.
20    READ(M1,1000) ICOD,(V(I),I=1,L7)
1000  FORMAT(10I1,7F10.0)
      IF(M.GE.0) WRITE(MP,2020) ICOD,(V(I),I=1,L7)
2020  FORMAT(' >>>>>',10I1,7E12.5)
C------- CHECK FOR A BLANK CARD
      J=0
      DO 30 I=1,10
30    J=J+ICOD(I)
      IF(J.EQ.0) GO TO 110
C------- READ ADDITIONAL CARD IF REQUIRED
      I2=0
      DO 40 ID=1,NDLN
      IF(ICOD(ID).LT.2) GO TO 40
      I2=I2+1
      IF(I2.NE.L8) GO TO 40
      READ(M1,1010) (V(I),I=L8,NDLN)
1010  FORMAT(10X,7F10.0)
      IF(M.GE.0) WRITE(MP,2030) (V(I),I=L8,NDLN)
2030  FORMAT(' >>>>>',10X,7E12.5)
40    CONTINUE
C------- READ NODE CARDS
50    READ(M1,1020) (KV(IN),IN=1,L16)
```

```
1020  FORMAT(16I5)
      IF(M.GE.0) WRITE(MP,2040) (KV(IN),IN=1,L16)
2040  FORMAT(' >>>>>',10X,16I5)
C------- FORM NEQ
      DO 100 IN=1,L16
      I2=KV(IN)
C------- END OF GROUP OF B.C. OR END OF NODES OR ANALYSIS OF A NODE
      IF(I2) 20,20,60
60    IF(I2.GT.NNT) CALL ERREUR(32,I2,NNT,1)
      I1=KDLNC(I2)
      IDN=KDLNC(I2+1)-I1
C-------  GENERATE VDIMP, PUT IT IN KNEQ (THE PRESCRIBED D.O.F. ADDRESS)
      IV=0
      DO 90 ID=1,IDN
      I1=I1+1
      IC=ICOD(ID)-1
      IF(IC) 90,70,80
70    NCLT=NCLT+1
      VDIMP(NCLT)=ZERO
      NCLZ=NCLZ+1
      KNEQ(I1)=-NCLT
      GO TO 90
80    NCLT=NCLT+1
      IV=IV+1
      VDIMP(NCLT)=V(IV)
      NCLNZ=NCLNZ+1
      KNEQ(I1)=-NCLT
90    CONTINUE
100   CONTINUE
C------- ADDITIONAL CARD OF NODE NUMBERS
      GO TO 50
C------- GENERATE EQUATION NUMBERS IN NEQ
110   I1=0
      DO 150 IN=1,NNT
      ID=KDLNC(IN)
120   ID=ID+1
      IF(ID.GT.KDLNC(IN+1)) GO TO 150
      IF(KNEQ(ID)) 120,130,120
130   I1=I1+1
      KNEQ(ID)=I1
      GO TO 120
150   CONTINUE
      NEQ=I1
C------- OUTPUT
      IF(M.LT.0) GO TO 170
      WRITE(MP,2050) NNT,NDLT,NEQ,NCLNZ,NCLZ,NCLT
2050  FORMAT(//
     1  15X,'TOTAL NUMBER OF NODES                          (NNT)=',I5/
     2  15X,'TOTAL NUMBER OF D.O.F.                        (NDLT)=',I5/
     3  15X,'NUMBER OF EQUATIONS TO BE SOLVED               (NEQ)=',I5/
     4  15X,'NUMBER OF PRESCRIBED NON ZERO D.O.F.         (NCLNZ)=',I5/
```

```
    5  15X,'NUMBER OF PRESCRIBED ZERO D.O.F.              (NCLZ)=',I5/
    6  15X,'TOTAL NUMBER OF PRESCRIBED D.O.F.             (NCLT)=',I5/)
       IF(M.GE.2.AND.NCLT.GT.0) WRITE(MP,2060)(VDIMP(I),I=1,NCLT)
2060   FORMAT(//' PRESCRIBED VALUES  (VDIMP)'//(10X,10E12.5))
       WRITE(MP,2070)
2070   FORMAT(//' NODAL COORDINATES ARRAY'//
    1  '   NO  D.L.',5X,'X',12X,'Y',12X,'Z',10X,'EQUATION NUMBER
    2(NEQ)'/)
       I2=0
       DO 160 IN=1,NNT
       I1=I2+1
       I2=I2+NDIM
       ID1=KDLNC(IN)+1
       ID2=KDLNC(IN+1)
       ID=ID2-ID1+1
       IF(ID2.LT.ID1) ID2=ID1
       X1=VCORG(I1)
       IF(NDIM.GE.2) X2=VCORG(I1+1)
       IF(NDIM.GE.3) X3=VCORG(I1+2)
160    WRITE(MP,2080) IN,ID,X1,X2,X3,(KNEQ(I),I=ID1,ID2)
2080   FORMAT(1X,2I5,3E12.5,10X,10I6)
170    RETURN
       END
```

```
      SUBROUTINE BLPRND
C=====================================================================
C     TO CALL BLOCK 'PRND'
C     TO READ NODAL PROPERTIES
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/COOR/NDIM,NNT,NNULL(2),FNULL(3)
      COMMON/PRND/NPRN
      COMMON/ES/M,MR,MP,M1,MDUMMY(9)
      COMMON/LOC/LXX(4),LPRNG,LDUMMY(20)
      COMMON VA(1)
      DATA TBL/'PRNG'/
C---------------------------------------------------------------------
      IF(M1.EQ.0) M1=MR
      READ(M1,1000) NPRN
1000  FORMAT(I5)
      WRITE(MP,2000) M,NPRN
2000  FORMAT(//' INPUT OF NODAL PROPERTIES (M=',I2,')'/' ',30('=')/
     1  15X,'NUMBER OF PROPERTIES PER NODE     (NPRN)=',I5)
      IF(LPRNG.EQ.1) CALL ESPACE(NNT*NPRN,1,TBL,LPRNG)
      CALL EXPRND(VA(LPRNG))
      RETURN
      END


      SUBROUTINE EXPRND(VPRNG)
C=====================================================================
C     TO EXECUTE BLOCK 'PRND'
C     READ NODAL PROPERTIES
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNT,NNULL(2),FNULL(3)
      COMMON/PRND/NPRN
      COMMON/ES/M,MR,MP,M1,MDUMMY(9)
      DIMENSION VPRNG(*)
C---------------------------------------------------------------------
C-------  READ PROPERTIES NODEWISE
      I1=NNT*NPRN
      READ(M1,1000)(VPRNG(I),I=1,I1)
1000  FORMAT(8F10.0)
      IF(M.GE.0) WRITE(MP,2000) (VPRNG(I),I=1,I1)
2000  FORMAT(//' CARDS OF NODAL PROPERTIES'/ (' >>>>>',8E12.5))
      RETURN
      END
```

```
      SUBROUTINE BLPREL
C=====================================================================
C     TO CALL BLOCK 'PREL'
C     TO READ ELEMENT PROPERTIES
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/PREL/NGPE,NPRE
      COMMON/ES/M,MR,MP,M1,MDUMMY(9)
      COMMON/LOC/LXX(5),LPREG,LDUMMY(19)
      COMMON/TRVL/IN(2),RDUMMY(520),NULL
      COMMON VA(1)
      DIMENSION TBL(2)
C
C+++     THIS IS COMMENTED OUT BECAUSE OF THE MS FORTRAN COMP-
C+++     ILER BUG SHICH WILL NOT INITIALIZE $LARGE ARRAYS
C+++     THIS ARRAY IS NOW INITIALIZED BY A CALL TO INITBL WHICH
C+++     EXISTS SOLELY TO INITIALIZE TABLE NAMES.
C
C     DATA TBL/'PREG','V   '/
C
C        HERE IS THE CALL TO GET AROUND THE COMPILER BUG
C
      CALL INITBL(TBL,'PREL')
C
C+++     ALL THIS WAS SIMPLY TO GET AROUND THE MICROSOFT
C+++     COMPILER BUG
C
C---------------------------------------------------------------------
      IF(M1.EQ.0) M1=MR
C------  READ NUMBER OF GROUPS AND PROPERTIES PER GROUP
      READ(M1,1000) IN
1000  FORMAT(2I5)
      IF(IN(1).GT.0) NGPE=IN(1)
      IF(IN(2).GT.0) NPRE=IN(2)
      WRITE(MP,2000) M,NGPE,NPRE
2000  FORMAT(//' INPUT OF ELEMENT PROPERTIES (M=',I2,')'/' ',
     1  35('=')/15X,'NUMBER OF GROUPS OF PROPERTIES   (NGPE)=',I5/
     2  15X,'NUMBER OF PROPERTIES PER GROUP   (NPRE)=',I5)
      IF(LPREG.EQ.1) CALL ESPACE(NGPE*NPRE,1,TBL(1),LPREG)
      CALL ESPACE(NPRE,1,TBL(2),L1)
      CALL EXPREL(VA(LPREG),VA(L1))
      CALL VIDE(L1,1,TBL(2))
      RETURN
      END
```

```
      SUBROUTINE EXPREL(VPREG,V1)
C=====================================================================
C     TO EXECUTE BLOCK 'PREL'
C     READ ELEMENT PROPERTIES
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/PREL/NGPE,NPRE
      COMMON/ES/M,MR,MP,M1,MDUMMY(9)
      DIMENSION VPREG(*),V1(*)
C---------------------------------------------------------------------
      IF(M.GE.0) WRITE(MP,2000)
2000  FORMAT(//' CARDS OF ELEMENT PROPERTIES'/)
C------- READ A GROUP
      I1=MIN0(7,NPRE)
      J=1
10    READ(M1,1000) IGPE,(V1(I),I=1,I1)
1000  FORMAT(I5,7F10.0)
      IF(M.GE.0) WRITE(MP,2010) IGPE,(V1(I),I=1,I1)
2010  FORMAT(' >>>>>',I5,7E12.5)
      IF(IGPE.LE.0) GO TO 40
      IF(IGPE.GT.NGPE) CALL ERREUR(41,IGPE,NGPE,1)
      IF(NPRE.LE.7) GO TO 20
C------- READ THE PROPERTIES
      READ(M1,1010) (V1(I),I=8,NPRE)
1010  FORMAT(5X,7F10.0)
      IF(M.GE.0) WRITE(MP,2020) (V1(I),I=8,NPRE)
2020  FORMAT(' >>>>>',5X,7E12.5)
20    DO 30 I=1,NPRE
      VPREG(J)=V1(I)
30    J=J+1
      GO TO 10
40    RETURN
      END
```

```
      SUBROUTINE BLELEM
C=================================================================
C     TO CALL BLOCK 'ELEM'
C     TO READ ELEMENT DATA
C=================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/COOR/NDIM,NNT,NDLN,NNULL,FNULL(3)
      COMMON/PRND/NPRN
      COMMON/PREL/NGPE,NPRE
      COMMON/ELEM/NELT,NNEL,NTPE,NGRE,ME,NIDENT,NPG
      COMMON/ASSE/NSYM,NKG,MFILLR(2)
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/ES/M,MR,MP,M1,M2,MDUMMY(3)
      COMMON/LOC/LCORG,LDLNC,LNEQ,LDIMP,LPRNG,LPREG,LLD,LLOCE,LCORE,LNE,
     1  LPRNE,LPREE,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,LRES,LDUMMY(5)
      COMMON VA(1)
      DIMENSION TBL(6),IN(6)
C
C+++     THIS IS COMMENTED OUT BECAUSE OF THE MS FORTRAN COMP-
C+++     ILER BUG SHICH WILL NOT INITIALIZE $LARGE ARRAYS
C+++     THIS ARRAY IS NOW INITIALIZED BY A CALL TO INITBL WHICH
C+++     EXISTS SOLELY TO INITIALIZE TABLE NAMES.
C
C     DATA TBL/'LD  ','LOCE','CORE','NE  ','PRNE','PREE'/
C
C        HERE IS THE CALL TO GET AROUND THE COMPILER BUG
C
      CALL INITBL(TBL,'ELEM')
C
C+++     ALL THIS WAS SIMPLY TO GET AROUND THE MICROSOFT
C+++     COMPILER BUG
C
C-----------------------------------------------------------------
      IF(M1.EQ.0) M1=MR
      IF(M2.EQ.0) M2=ME
      READ(M1,1000)IN
1000  FORMAT(6I5)
      IF(IN(1).GT.0) NELT=IN(1)
      IF(IN(2).GT.0) NNEL=IN(2)
      IF(IN(3).GT.0) NTPE=IN(3)
      IF(IN(4).GT.0) NGRE=IN(4)
      IF(IN(5).NE.0) NSYM=1
      IF(IN(6).NE.0) NIDENT=1
      WRITE(MP,2000) M,NELT,NNEL,NTPE,NGRE,NSYM,NIDENT
2000  FORMAT(//' INPUT OF ELEMENTS (M=',I2,')'/' ',20('=')/
     1  15X,'MAX. NUMBER OF ELEMENTS                (NELT)=',I5/
     2  15X,'MAX. NUMBER OF NODES PER ELEMENT       (NNEL)=',I5/
     3  15X,'DEFAULT ELEMENT TYPE                   (NTPE)=',I5/
     4  15X,'NUMBER OF GROUPS OF ELEMENTS           (NGRE)=',I5/
```

```
     5  15X,'INDEX FOR NON SYMMETRIC PROBLEM     (NSYM)=',I5/
     6  15X,'INDEX FOR IDENTICAL ELEMENTS      (NIDENT)=',I5/)
      IF(LLD.EQ.1) CALL ESPACE(NEQ+1,0,TBL(1),LLD)
      IF(LLOCE.EQ.1) CALL ESPACE(NNEL*NDLN,0,TBL(2),LLOCE)
      IF(LCORE.EQ.1) CALL ESPACE(NNEL*NDIM,1,TBL(3),LCORE)
      IF(LNE.EQ.1) CALL ESPACE(NNEL,0,TBL(4),LNE)
      IF(NPRN.GT.0.AND.LPRNE.EQ.1) CALL ESPACE(NNEL*NPRN,1,TBL(5),LPRNE)
      IF(NPRE.GT.0.AND.LPREE.EQ.1) CALL ESPACE(NPRE,1,TBL(6),LPREE)
      CALL EXELEM(VA(LCORG),VA(LDLNC),VA(LPRNG),VA(LPREG),VA(LLOCE),
     1          VA(LCORE),VA(LNE),VA(LPRNE),VA(LPREE),VA(LNEQ),VA(LLD))
      WRITE(MP,2010) NKG,NPG
2010  FORMAT(15X,'LENGTH OF A TRIANGLE IN KG          (NKG)=',I10/
     1       15X,'NUMBER OF INTEGRATION POINTS        (NPG)=',I10/)
      RETURN
      END


      SUBROUTINE EXELEM(VCORG,KDLNC,VPRNG,VPREG,KLOCE,VCORE,KNE,VPRNE,
     1                  VPREE,KNEQ,KLD)
C======================================================================
C     TO EXECUTE BLOCK 'ELEM'
C     READ ELEMENTS DATA
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNT,NNULL(2),FNULL(3)
      COMMON/PRND/NPRN
      COMMON/PREL/NGPE,NPRE
      COMMON/ELEM/NELT,NNEL,NTPE,NGRE,ME,NIDENT,NPG
      COMMON/ASSE/NSYM,NKG,NKE,NDLE
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,IPG
     1  ,ICODE,IDLEO,INELO,IPGO
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/ES/M,MR,MP,M1,M2,MDUMMY(8)
      DIMENSION VCORG(*),KDLNC(*),VPRNG(*),VPREG(*),KLOCE(*),VCORE(*),
     1          KNE(*),VPRNE(*),VPREE(*),KNEQ(*),KLD(*)
      DATA I10/10/,I15/15/
C----------------------------------------------------------------------
C------- INITIALIZE
      OPEN(M2,FILE='$$002.DAT',STATUS='NEW',FORM='UNFORMATTED')
      NDLE=0
      IELT=0
      NPG=0
      REWIND M2
      IF(M.GT.0) WRITE(MP,2000)
2000  FORMAT(//' ELEMENTS CARDS'/)
C------- READ AN ELEMENT CARD
10    READ(M1,1000) IEL,IGEN,INCR,ITPE,IGPE,IGRE,(KNE(IN),IN=1,I10)
1000  FORMAT(16I5)
      IF(M.GT.0) WRITE(MP,2010) IEL,IGEN,INCR,ITPE,IGPE,IGRE,
     1                          (KNE(IN),IN=1,I10)
2010  FORMAT(' >>>>>',16I5)
```

```
      IF(IEL) 80,80,20
C------ NUMBER OF NODES AND ADDITIONNAL CARDS AS REQUIRED
20    INEL=0
      I1=1
      I2=I10
30    DO 40 IN=I1,I2
      IF(KNE(IN).EQ.0) GO TO 50
      INEL=INEL+1
40    CONTINUE
      I1=I2+1
      I2=I1+I15
      READ(M1,1000) (KNE(IN),IN=I1,I2)
      IF(M.GT.0) WRITE(MP,2010) (KNE(IN),IN=I1,I2)
      GO TO 30
C------- CHECKING
50    IF(INEL.GT.NNEL) CALL ERREUR(51,INEL,NNEL,1)
      IF(INCR.EQ.0) INCR=1
      IF(ITPE.EQ.0) ITPE=NTPE
      IF(IGPE.GT.NGPE) CALL ERREUR(53,IGPE,NGPE,1)
      IF(IGPE.EQ.0) IGPE=1
      IF(IGRE.GT.NGRE) CALL ERREUR(54,IGRE,NGRE,1)
C------ ELEMENT GENERATION
      IF(IGEN.EQ.0) IGEN=1
      DO 70 IE=1,IGEN
      IF(IEL.GT.NELT) CALL ERREUR(55,IEL,NELT,1)
C------- GENERATE KLOCE AND UPDATE KLD
      CALL LOCELD(KDLNC,KNE,KNEQ,KLOCE,KLD)
C------- GENERATE ELEMENT COORDINATES AND PROPERTIES
      CALL XTRELM(IGPE,VCORG,VPRNG,VPREG,KNE,VCORE,VPRNE,VPREE)
C------- CHECK ELEMENT NODE NUMBERS AND D.O.F.
      IPGO=0
      ICODE=1
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
      IF(INEL.EQ.INELO.AND.IDLE.EQ.IDLEO) GO TO 55
      WRITE(MP,2020) IEL,INEL,INELO,IDLE,IDLEO
2020  FORMAT(' ** ELEMENT',I5,' INCONSISTENT'/5X,'INEL=',I4,' INELO=',I5
     1/  5X,'IDLE=',I5,' IDLEO=',I5)
C------- UPDATE TOTAL NUMBER OF INTEGRATION POINTS
55    NPG=NPG+IPGO
C------- STORE ON ELEMENT FILE
      CALL WRELEM(M2,KLOCE,VCORE,VPRNE,VPREE,KNE)
      IELT=IELT+1
C------- PRINT ELEMENT CHARACTERISTICS
      CALL PRELEM(KLOCE,VCORE,VPRNE,VPREE,KNE)
C------- NEXT ELEMENT TO BE GENERATED OR READ
      DO 60 IN=1,INEL
60    KNE(IN)=KNE(IN)+INCR
      IF(IDLE.GT.NDLE) NDLE=IDLE
70    IEL=IEL+1
      GO TO 10
C------- CHECK IF TOTAL NUMBER OF ELEMENT IS EXCEEDED
```

```
80    IF(IELT.NE.NELT) CALL ERREUR(57,IELT,NELT,1)
C------- PRINT BAND HEIGHTS
      IMA=0
      IMO=0
      I1=NEQ+1
      DO 90 I=2,I1
      J=KLD(I)
      IF(J.GT.IMA)IMA=J
90    IMO=IMO+J
      C=IMO
      C=C/NEQ
      WRITE(MP,2030) C,IMA
2030  FORMAT(/15X,'MEAN BAND HEIGHT=',F8.1,' MAXIMUM=',I5)
      IF(M.GE.2) WRITE(MP,2040) (KLD(I),I=1,I1)
2040  FORMAT(//' TABLE OF BAND HEIGHTS'/(10X,20I5))
C------- TRANSFORM KLD INTO ADDRESSES OF COLUMN TOP TERM
      IF(NSYM.EQ.0) NKE=(NDLE*(NDLE+1))/2
      IF(NSYM.EQ.1) NKE=NDLE*NDLE
      KLD(1)=1
      DO 100 ID=2,I1
100   KLD(ID)=KLD(ID-1)+KLD(ID)
      NKG=KLD(I1)-1
      IF(M.GE.2) WRITE(MP,2050) (KLD(ID),ID=1,I1)
2050  FORMAT(//' TABLE OF ADDRESSES OF COLUMN TOP TERMS  (LD)'/
     1        (10X,20I6))
      RETURN
      END
```

```
$LARGE
$NOFLOATCALLS
      SUBROUTINE LOCELD(KDLNC,KNE,KNEQ,KLOCE,KLD)
C=====================================================================
C     TO FORM THE ELEMENT LOCALIZATION TABLE (LOCE)
C     AND UPDATE COLUMN HEIGHTS FOR A GIVEN ELEMENT
C=====================================================================
      REAL*8 FNULL
      COMMON/COOR/NDIM,NNT,NNULL(2),FNULL(3)
      COMMON/RGDT/NUL(4),IDLE,NUL1(3),INEL,IDUMMY(6)
      DIMENSION KDLNC(*),KNE(*),KNEQ(*),KLOCE(*),KLD(*)
      DATA NDLMAX/32000/
C---------------------------------------------------------------------
C------- GENERATE KLOCE FROM KNEQ
      IDLE=0
      LOCMIN=NDLMAX
      DO 20 IN=1,INEL
      INN=KNE(IN)
      IF(INN.GT.NNT) CALL ERREUR(56,INN,NNT,1)
      IEQ=KDLNC(INN)
      IEQ1=KDLNC(INN+1)
   10 IF(IEQ.GE.IEQ1) GO TO 20
      IEQ=IEQ+1
      IDLE=IDLE+1
      J=KNEQ(IEQ)
      KLOCE(IDLE)=J
      IF(J.LT.LOCMIN.AND.J.GT.0) LOCMIN=J
      GO TO 10
   20 CONTINUE
C------- UPDATE TABLE OF COLUMN HEIGHTS (KLD)
      DO 30 ID=1,IDLE
      J=KLOCE(ID)
      IF(J.LE.0) GO TO 30
      IH=J-LOCMIN
      IF(IH.GT.KLD(J+1))KLD(J+1)=IH
   30 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE XTRELM(IGPE,VCORG,VPRNG,VPREG,KNE,VCORE,VPRNE,VPREE)
C======================================================================
C     TO GENERATE ELEMENT COORDINATES AND PROPERTIES FROM
C     GLOBAL ARRAYS
C     (IGPE: GROUP NUMBER FOR ELEMENT PROPERTIES)
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNULL(3),FNULL(3)
      COMMON/PRND/NPRN
      COMMON/PREL/NGPE,NPRE
      COMMON/RGDT/NUL(5),ICE,IPRNE,IPREE,INEL,IDUMMY(6)
      DIMENSION VCORG(*),VPRNG(*),VPREG(*),KNE(*),VCORE(*),
     1  VPRNE(*),VPREE(*)
C----------------------------------------------------------------------
C------- GENERATE ELEMENT COORDINATES
      IPRNE=0
      ICE=0
      DO 30 IN=1,INEL
      IC=(KNE(IN)-1)*NDIM
      DO 10 I=1,NDIM
      ICE=ICE+1
      IC=IC+1
   10 VCORE(ICE)=VCORG(IC)
C------- GENERATE ELEMENT NODAL PROPERTIES
      IF(NPRN.EQ.0) GO TO 30
      IC=(KNE(IN)-1)*NPRN
      DO 20 I=1,NPRN
      IPRNE=IPRNE+1
      IC=IC+1
   20 VPRNE(IPRNE)=VPRNG(IC)
   30 CONTINUE
C------- GENERATE ELEMENT PROPERTIES
      IPREE=0
      IF(NPRE.EQ.0) GO TO 50
      IC=(IGPE-1)*NPRE
      DO 40 I=1,NPRE
      IPREE=IPREE+1
      IC=IC+1
   40 VPREE(IPREE)=VPREG(IC)
50    RETURN
      END
```

```
      SUBROUTINE PRELEM(KLOCE,VCORE,VPRNE,VPREE,KNE)
C======================================================================
C     PRINT DATA DEFINING AN ELEMENT
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/PRND/NPRN
      COMMON/PREL/NGPE,NPRE
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,NULL(6)
      COMMON/ES/M,MR,MP,MDUMMY(10)
      DIMENSION KLOCE(*),VCORE(*),VPRNE(*),VPREE(*),KNE(*)
C----------------------------------------------------------------------
      IF(M.GE.0) WRITE(MP,2000) IEL,ITPE,INEL,IDLE,IPRNE,IPREE,IGRE
2000  FORMAT(10X,'ELEMENT:',I5,' TYPE:',I2,' N.P.:',I2,' D.O.F.:',
     1  I3,' N. PROP:',I3,' EL. PROP:',I3,' GROUP:',I3)
      IF(M.GE.0) WRITE(MP,2010) (KNE(I),I=1,INEL)
2010  FORMAT(15X,'CONNECTIVITY (NE)',20I5/(32X,20I5))
      IF(M.LT.1) GO TO 10
      WRITE(MP,2020) (KLOCE(I),I=1,IDLE)
2020  FORMAT(15X,'LOCALIZATN (LOCE)',20I5/(32X,20I5))
      WRITE(MP,2030) (VCORE(I),I=1,ICE)
2030  FORMAT(15X,'COORDINATES(CORE)',8E12.5/(32X,8E12.5))
      IF(NPRN.GT.0) WRITE(MP,2040) (VPRNE(I),I=1,IPRNE)
2040  FORMAT(15X,'NOD.PROP.  (PRNE)',8E12.5/(32X,8E12.5))
      IF(IPREE.GT.0) WRITE(MP,2050) (VPREE(I),I=1,IPREE)
2050  FORMAT(15X,'ELEM. PROP.(PREE)',8E12.5/(32X,8E12.5))
10    RETURN
      END


      SUBROUTINE WRELEM(ME,KLOCE,VCORE,VPRNE,VPREE,KNE)
C======================================================================
C     WRITE ELEMENT PROPERTIES ON FILE ME
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,NULL(6)
      DIMENSION KLOCE(*),VCORE(*),VPRNE(*),VPREE(*),KNE(*)
C----------------------------------------------------------------------
      IPRNE1=IPRNE
      IF(IPRNE1.EQ.0) IPRNE1=1
      IPREE1=IPREE
      IF(IPREE1.EQ.0) IPREE1=1
      WRITE(ME)IEL,ITPE,IGRE,IDLE,ICE,IPRNE1,IPREE1,INEL,
     1          (KLOCE(I),I=1,IDLE),(VCORE(I),I=1,ICE),
     2          (VPRNE(I),I=1,IPRNE1),(VPREE(I),I=1,IPREE1),
     3          (KNE(I),I=1,INEL)
      RETURN
      END
```

```
      SUBROUTINE RDELEM(ME,KLOCE,VCORE,VPRNE,VPREE,KNE)
C======================================================================
C     READ ELEMENT PROPERTIES FROM FILE ME
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,NULL(6)
      DIMENSION KLOCE(*),VCORE(*),VPRNE(*),VPREE(*),KNE(*)
C----------------------------------------------------------------------
      READ(ME) IEL,ITPE,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,
     1         (KLOCE(I),I=1,IDLE),(VCORE(I),I=1,ICE),
     2         (VPRNE(I),I=1,IPRNE),(VPREE(I),I=1,IPREE),
     3         (KNE(I),I=1,INEL)
      RETURN
      END


      SUBROUTINE BLSOLC
C======================================================================
C     TO CALL BLOCK 'SOLC'
C     TO READ CONCENTRATED LOADS
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/ES/M,MR,MP,M1,MDUMMY(9)
      COMMON/LOC/LCORG,LDLNC,LNEQ,LXX(15),LFG,LDUMMY(6)
      COMMON VA(1)
      DATA TBL/'FG  '/
C----------------------------------------------------------------------
      IF(M1.EQ.0) M1=MR
      WRITE(MP,2000) M
2000  FORMAT(//' INPUT OF CONCENTRADED LOADS (M=',I2,')'/' ',
     1  39('='))
      IF(LFG.EQ.1) CALL ESPACE(NEQ,1,TBL,LFG)
      CALL EXSOLC(VA(LFG),VA(LDLNC),VA(LNEQ))
      RETURN
      END
```

```
      SUBROUTINE EXSOLC(VFG,KDLNC,KNEQ)
C==================================================================
C     TO EXECUTE BLOCK 'SOLC'
C     READ CONCENTRATED LOADS
C==================================================================
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/COOR/NDIM,NNT,NDLN,NNULL,FNULL(3)
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/ES/M,MR,MP,M1,MDUMMY(9)
      COMMON/TRVL/KV(16),V(14),RDUMMY(499),NULL
      DIMENSION VFG(*),KDLNC(*),KNEQ(*)
      DATA L16/16/
C------------------------------------------------------------------
C------- READ DATA
      IF(M.GE.0)WRITE(MP,2000)
2000  FORMAT(//' CARDS OF NODAL LOADS'//)
      I0=MIN0(7,NDLN)
10    READ(M1,1000) IG,(V(I),I=1,I0)
1000  FORMAT(I5,7F10.0)
      IF(NDLN.GT.7) READ(M1,1005) (V(I),I=8,NDLN)
1005  FORMAT(5X,7F10.0)
      IF(M.GE.0)WRITE(MP,2010)IG,(V(I),I=1,NDLN)
2010  FORMAT(' >>>>>',I5,7E12.5/(' >>>>>',5X,7E12.5))
      IF(IG.LE.0) GO TO 60
20    READ(M1,1010)(KV(I),I=1,L16)
1010  FORMAT(16I5)
      IF(M.GE.0)WRITE(MP,2020)(KV(I),I=1,L16)
2020  FORMAT(' >>>>>',16I5)
C----- DECODE NODAL DATA
      DO 50 IN=1,L16
      I1=KV(IN)
      IF(I1.GT.NNT) CALL ERREUR(61,I1,NNT,1)
      IF(I1)10,10,30
30    ID1=KDLNC(I1)+1
      ID2=KDLNC(I1+1)
      J=0
      DO 50 ID=ID1,ID2
      J=J+1
      IEQ=KNEQ(ID)
      IF(IEQ)50,50,40
40    VFG(IEQ)=VFG(IEQ)+V(J)
50    CONTINUE
      GO TO 20
C----- OUTPUT
60    IF(M.GE.1)WRITE(MP,2030)(VFG(I),I=1,NEQ)
2030  FORMAT(//' TOTAL LOAD VECTOR'/(10X,10E12.5))
      RETURN
      END
```

```
      SUBROUTINE BLSOLR
C=======================================================================
C     TO CALL BLOCK 'SOLR'
C     TO ASSEMBLE DISTRIBUTED LOADS (ELEMENT FUNCTION 7)
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/COOR/NDIM,NNT,NDLN,NDLT,FNULL(3)
      COMMON/ELEM/NULL(4),ME,MNULL(2)
      COMMON/ASSE/NSYM,NKG,NKE,NDLE
      COMMON/RESO/NEQ,NRES,MRES
      COMMON/ES/M,MR,MP,M1,M2,MDUMMY(8)
      COMMON/LOC/LCORG,LDLNC,LNEQ,LDIMP,LPRNG,LPREG,LLD,LLOCE,LCORE,LNE,
     1 LPRNE,LPREE,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,LRES,LDLG,LDUMMY(4)
      COMMON VA(1)
      DIMENSION TBL(8)
C
C+++     THIS IS COMMENTED OUT BECAUSE OF THE MS FORTRAN COMP-
C+++     ILER BUG WHICH WILL NOT INITIALIZE $LARGE ARRAYS
C+++     THIS ARRAY IS NOW INITIALIZED BY A CALL TO INITBL WHICH
C+++     EXISTS SOLELY TO INITIALIZE TABLE NAMES.
C
C     DATA TBL/'FG  ','KE  ','FE  ','DLE ','KGS ','KGD ','KGI ',
C    1  'RES '/
C
C        HERE IS THE CALL TO GET AROUND THE COMPILER BUG
C
      CALL INITBL(TBL,'SOLR')
C
C+++     ALL THIS WAS SIMPLY TO GET AROUND THE MICROSOFT
C+++     COMPILER BUG
C
C-----------------------------------------------------------------------
      IF(M1.EQ.0) M1=MR
      IF(M2.EQ.0) M2=ME
      WRITE(MP,2000) M
2000  FORMAT(//' ASSEMBLING OF DISTRIBUTED LOADS (M=',I2,')'/
     1  1X,40('=')/)
      IF(LFG.EQ.1) CALL ESPACE(NEQ,1,TBL(1),LFG)
      IF(LKE.EQ.1) CALL ESPACE(NKE,1,TBL(2),LKE)
      IF(LFE.EQ.1) CALL ESPACE(NDLE,1,TBL(3),LFE)
      IF(LDLE.EQ.1) CALL ESPACE(NDLE,1,TBL(4),LDLE)
      IF(LKGS.EQ.1) CALL ESPACE(NKG,1,TBL(5),LKGS)
      IF(LKGD.EQ.1) CALL ESPACE(NEQ,1,TBL(6),LKGD)
      IF(NSYM.EQ.1.AND.LKGI.EQ.1) CALL ESPACE(NKG,1,TBL(7),LKGI)
      IF(LRES.EQ.1) CALL ESPACE(NDLT,1,TBL(8),LRES)
      CALL EXSOLR(VA(LLD),VA(LDIMP),VA(LLOCE),VA(LCORE),VA(LPRNE),
     1            VA(LPREE),VA(LNE),VA(LKE),VA(LFE),VA(LKGS),VA(LKGD),
     2            VA(LKGI),VA(LFG),VA(LCORG),VA(LDLNC),VA(LNEQ),
     3            VA(LRES),VA(LDLE))
      RETURN
      END
```

```
      SUBROUTINE EXSOLR(KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,
     1         VKGS,VKGD,VKGI,VFG,VCORG,KDLNC,KNEQ,VRES,VDLE)
C===========================================================================
C     TO EXECUTE BLOCK 'SOLR'
C     ASSEMBLE DISTRIBUTED LOADS (ELEMENT FUNCTION 7)
C===========================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ASSE/NSYM,NKG,NKE,NDLE
      COMMON/RESO/NEQ,NRES,NFILLR
      COMMON/ES/M,MR,MP,M1,M2,MDUMMY(8)
      DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),VPRNE(*),VPREE(*),
     1  KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),VCORG(*),
     2  KDLNC(*),KNEQ(*),VRES(*),VDLE(*)
C---------------------------------------------------------------------------
C-----  ASSEMBLE FG
      CALL ASFG(KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,VKGS,
     1  VKGD,VKGI,VFG,VDLE,VRES)
C-------  OUTPUT
      IF(M.GE.1) WRITE(MP,2000) (VFG(I),I=1,NEQ)
2000  FORMAT(/' GLOBAL LOAD VECTOR  (FG)'/(1X,10E12.5))
      RETURN
      END
```

```
      SUBROUTINE ASFG(KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,
     1  VKGS,VKGD,VKGI,VFG,VDLE,VRES)
C=======================================================================
C     ASSEMBLING DISTRIBUTED LOADS IN FG
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ELEM/NELT,NNEL,NTPE,NGRE,ME,NIDENT,MNULL
      COMMON/ASSE/NSYM,MFILLR(3)
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,IPG
     1  ,ICOD,NULL(3)
      COMMON/ES/M,MR,MP,M1,M2,MDUMMY(8)
      DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),VPRNE(*),VPREE(*),
     1  KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),VDLE(*),
     2  VRES(*)
C-----------------------------------------------------------------------
C-------  REWIND ELEMENT FILE M2
      REWIND M2
C-------  LOOP OVER THE ELEMENTS
      DO 20 IE=1,NELT
C-------  READ AN ELEMENT FROM FILE M2
      CALL RDELEM(M2,KLOCE,VCORE,VPRNE,VPREE,KNE)
C-------  EVALUATE INTERPOLATION FUNCTIONS IF REQUIRED
      IF(ITPE.EQ.ITPE1) GO TO 10
      ICOD=2
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C-------  EVALUATE ELEMENT VECTOR
10    ICOD=7
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C-------  PRINT ELEMENT VECTOR VFE
      IF(M.GE.2) WRITE(MP,2000) IEL,(VFE(I),I=1,IDLE)
2000  FORMAT(/' VECTOR (FE) , ELEMENT:',I5/(10X,10E12.5))
C-------  ASSEMBLE
      CALL ASSEL(0,1,IDLE,NSYM,KLOCE,KLD,VKE,VFE,VKGS,VKGD,VKGI,VFG)
20    ITPE1=ITPE
      RETURN
      END
```

```
      SUBROUTINE BLLINM
C=========================================================================
C     TO CALL BLOCK 'LINM'
C     ASSEMBLE AND SOLVE A LINEAR PROBLEM IN CORE
C=========================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/COOR/NDIM,NNT,NDLN,NDLT,FNULL(3)
      COMMON/ELEM/NULL(4),ME,MNULL(2)
      COMMON/ASSE/NSYM,NKG,NKE,NDLE
      COMMON/RESO/NEQ,NRES,MRES
      COMMON/ES/M,MR,MP,M1,M2,M3,MDUMMY(7)
      COMMON/LOC/LCORG,LDLNC,LNEQ,LDIMP,LPRNG,LPREG,LLD,LLOCE,LCORE,LNE,
     1  LPRNE,LPREE,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,LRES,LDLG,LDUMMY(4)
      COMMON VA(1)
      DIMENSION TBL(8)
C
C+++     THIS IS COMMENTED OUT BECAUSE OF THE MS FORTRAN COMP-
C+++     ILER BUG WHICH WILL NOT INITIALIZE $LARGE ARRAYS
C+++     THIS ARRAY IS NOW INITIALIZED BY A CALL TO INITBL WHICH
C+++     EXISTS SOLELY TO INITIALIZE TABLE NAMES.
C
C     DATA TBL/'KGS ','KGD ','KGI ','FG  ','KE  ','FE  ','RES ','DLE '/
C
C        HERE IS THE CALL TO GET AROUND THE COMPILER BUG
C
      CALL INITBL(TBL,'LINM')
C
C+++     ALL THIS WAS SIMPLY TO GET AROUND THE MICROSOFT
C+++     COMPILER BUG
C
C-------------------------------------------------------------------------
      IF(M1.EQ.0) M1=MR
      IF(M2.EQ.0) M2=ME
      IF(M3.EQ.0) M3=MRES
      READ(M1,1000) IN
1000  FORMAT(1I5)
      IF(IN.NE.0) NRES=1
      WRITE(MP,2000) M,NRES
2000  FORMAT(//' ASSEMBLING AND LINEAR SOLUTION (M=',I2,')'/' ',30('=')/
     1  15X,' INDEX FOR RESIDUAL COMPUTATION    (NRES)=',I5)
      IF(LKGS.EQ.1) CALL ESPACE(NKG,1,TBL(1),LKGS)
      IF(LKGD.EQ.1) CALL ESPACE(NEQ,1,TBL(2),LKGD)
      IF(NSYM.EQ.1.AND.LKGI.EQ.1) CALL ESPACE(NKG,1,TBL(3),LKGI)
      IF(LFG.EQ.1) CALL ESPACE(NEQ,1,TBL(4),LFG)
      IF(LKE.EQ.1) CALL ESPACE(NKE,1,TBL(5),LKE)
      IF(LFE.EQ.1) CALL ESPACE(NDLE,1,TBL(6),LFE)
      IF(LRES.EQ.1) CALL ESPACE(NDLT,1,TBL(7),LRES)
      IF(LDLE.EQ.1) CALL ESPACE(NDLE,1,TBL(8),LDLE)
      CALL EXLINM(VA(LLD),VA(LDIMP),VA(LLOCE),VA(LCORE),VA(LPRNE),
```

```
     1              VA(LPREE),VA(LNE),VA(LKE),VA(LFE),VA(LKGS),VA(LKGD),
     2              VA(LKGI),VA(LFG),VA(LCORG),VA(LDLNC),VA(LNEQ),
     3              VA(LRES),VA(LDLE))
      RETURN
      END


      SUBROUTINE EXLINM(KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,
     1       VKGS,VKGD,VKGI,VFG,VCORG,KDLNC,KNEQ,VRES,VDLE)
C======================================================================
C     TO EXECUTE BLOCK 'LINM'
C     ASSEMBLE AND SOLVE A LINEAR PROBLEM IN CORE .
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ASSE/NSYM,NKG,NKE,NDLE
      COMMON/RESO/NEQ,NRES,MRES
      COMMON/ES/M,MR,MP,M1,M2,M3,MDUMMY(7)
      DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),VPRNE(*),VPREE(*),
     1  KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),VCORG(*),
     2  KDLNC(*),KNEQ(*),VRES(*),VDLE(*)
C----------------------------------------------------------------------
      OPEN(M3,FILE='$$003.DAT',STATUS='NEW',FORM='UNFORMATTED')
      REWIND M3
C
C-------  ASSEMBLE KG
C
C-------  SAVE UNMODIFIED VECTOR FG (BY B.C.) ON FILE M3
      WRITE(M3) (VFG(I),I=1,NEQ)
      IF(M.GE.2) WRITE(MP,2000) (VFG(I),I=1,NEQ)
2000  FORMAT(/' GLOBAL LOAD VECTOR NON MODIFIED BY B.C. (FG)'
     1/(1X,10E12.5))
C-------  ASSEMBLE KG, MODIFY FG FOR THE B.C. AND SAVE THEM
      CALL ASKG(KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,VKGS,
     1  VKGD,VKGI,VFG,VDLE,VRES)
      WRITE(M3) (VFG(I),I=1,NEQ)
      WRITE(M3) (VKGS(I),I=1,NKG),(VKGD(I),I=1,NEQ)
      IF(NSYM.EQ.1) WRITE(M3) (VKGI(I),I=1,NKG)
C-------  PRINT KG AND FG
      IF(M.LT.2) GO TO 20
      WRITE(MP,2005) (VKGS(I),I=1,NKG)
2005  FORMAT(/' GLOBAL MATRIX (KG)'/'    UPPER TRIANGLE'/
     1  (1X,10E12.5))
      WRITE(MP,2010) (VKGD(I),I=1,NEQ)
2010  FORMAT('    DIAGONAL'/(1X,10E12.5))
      IF(NSYM.EQ.1) WRITE(MP,2020) (VKGI(I),I=1,NKG)
2020  FORMAT('    LOWER TRIANGLE'/(1X,10E12.5))
      WRITE(MP,2030) (VFG(I),I=1,NEQ)
2030  FORMAT(/' GLOBAL LOAD VECTOR MODIFIED BY THE B.C. (FG)'
     1 /(1X,10E12.5))
C
C-------  SOLVE
```

```
C
20    CALL SOL(VKGS,VKGD,VKGI,VFG,KLD,NEQ,MP,1,1,NSYM,ENERG)
      IF(NSYM.NE.1) WRITE(MP,2035) ENERG
2035  FORMAT(15X,'ENERGY    (ENERG)=',1E12.5)
      IF(M.LT.2) GO TO 30
      WRITE(MP,2040) (VKGS(I),I=1,NKG)
2040  FORMAT(/' TRIANGULARIZED MATRIX (KG)'/'    UPPER TRIANGLE'/
     1  (1X,10E12.5))
      WRITE(MP,2010) (VKGD(I),I=1,NEQ)
      IF(NSYM.EQ.1) WRITE(MP,2020) (VKGI(I),I=1,NKG)
C-------  PIVOTS OF KG AND DETERMINANT
30    CALL PRPVTS(VKGD)
C-------  EVALUATE AND PRINT RESIDUAL VECTOR K.U - F
      IF(NRES.EQ.1) CALL PRRESD(VKGS,VKGD,VKGI,VFG,KLD,VRES)
C-------  PRINT THE SOLUTION
      WRITE(MP,2050)
2050  FORMAT(//' SOLUTION'//)
      CALL PRSOL(KDLNC,VCORG,VDIMP,KNEQ,VFG)
C
C-------  EVALUATE AND PRINT GRADIENTS (STRESSES)
C
      CALL ASGRAD(KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,VKGS,
     1  VKGD,VKGI,VFG,VDLE,VRES)
C
C-------  EVALUATE AND PRINT EQUILIBRIUM RESIDUAL VECTOR
C
C-------  READ VECTOR FG AND CHANGE SIGN
      REWIND M3
      READ(M3) (VRES(I),I=1,NEQ)
      DO 40 I=1,NEQ
40    VRES(I)=-VRES(I)
C-------  ASSEMBLE THE RESIDUALS
      CALL ASRESD(1,1,KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,
     1  VKGS,VKGD,VKGI,VFG,VDLE,VRES,VRES(NEQ+1))
C-------  PRINT THE RESIDUALS
      WRITE(MP,2060)
2060  FORMAT(//' EQUILIBRIUM RESIDUALS AND REACTIONS'//)
      CALL PRSOL(KDLNC,VCORG,VRES(NEQ+1),KNEQ,VRES)
      RETURN
      END
```

```
      SUBROUTINE ASKG(KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,
     1  VKGS,VKGD,VKGI,VFG,VDLE,VRES)
C===========================================================================
C     TO ASSEMBLE GLOBAL MATRIX KG (ELEMENT FUNCTION 3)
C       TAKING INTO ACCOUNT OF NON ZERO PRESCRIBED D.O.F.
C===========================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COND/NCLT,NCLZ,NCLNZ
      COMMON/ELEM/NELT,NNEL,NTPE,NGRE,ME,NIDENT,MNULL
      COMMON/ASSE/NSYM,MFILLR(3)
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,IPG
     1  ,ICOD,NULL(3)
      COMMON/ES/M,MR,MP,M1,M2,MDUMMY(8)
      DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),VPRNE(*),VPREE(*),
     1  KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),VDLE(*),
     2  VRES(*),KEB(1)
C---------------------------------------------------------------------------
C------- REWIND ELEMENT FILE (M2)
      REWIND M2
C------- LOOP OVER THE ELEMENTS
      DO 30 IE=1,NELT
C------- READ AN ELEMENT ON FILE M2
      CALL RDELEM(M2,KLOCE,VCORE,VPRNE,VPREE,KNE)
C------- SKIP COMPUTATION IF IDENTICAL ELEMENTS ENCOUNTERED
      IF(NIDENT.EQ.1.AND.IE.GT.1) GO TO 20
C------- EVALUATE INTERPOLATION FUNCTIONS IF REQUIRED
      IF(ITPE.EQ.ITPE1) GO TO 10
      ICOD=2
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C------- FORM ELEMENT MATRIX
10    ICOD=3
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C------- PRINT ELEMENT MATRIX
      IF(M.LT.2) GO TO 20
      IF(NSYM.EQ.0) IKE=IDLE*(IDLE+1)/2
      IF(NSYM.EQ.1) IKE=IDLE*IDLE
      WRITE(MP,2000) IEL,(VKE(I),I=1,IKE)
2000  FORMAT(/' MATRIX (KE) , ELEMENT:',I5/(10X,10E12.5))
C------- MODIFY FG FOR NON ZERO PRESCRIBED D.O.F.
20    IF(NCLNZ.NE.0) CALL MODFG(IDLE,NSYM,KLOCE,VDIMP,VKE,VFG)
C------- ASSEMBLE
      CALL ASSEL(1,0,IDLE,NSYM,KLOCE,KLD,VKE,VFE,VKGS,VKGD,VKGI,VFG)
30    ITPE1=ITPE
      RETURN
      END
```

```
      SUBROUTINE ASGRAD(KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,
     1  VKGS,VKGD,VKGI,VFG,VDLE,VRES)
C=======================================================================
C     TO EVALUATE AND PRINT GRADIENTS (STRESSES) AT ELEMENT G.P.
C       (ELEMENT FUNCTION 8)
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ELEM/NELT,NNEL,NTPE,NGRE,ME,NIDENT,MNULL
      COMMON/ASSE/NSYM,MFILLR(3)
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,IPG
     1  ,ICOD,NULL(3)
      COMMON/ES/M,MR,MP,M1,M2,MDUMMY(8)
      DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),VPRNE(*),VPREE(*),
     1  KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),VDLE(*),
     2  VRES(*)
C-----------------------------------------------------------------------
C-------  REWIND ELEMENTS FILE (M2)
      REWIND M2
C-------  LOOP OVER THE ELEMENTS
      DO 20 IE=1,NELT
C-------  READ THE ELEMENT
      CALL RDELEM(M2,KLOCE,VCORE,VPRNE,VPREE,KNE)
C-------  EVALUATE INTERPOLATION FUNCTION IF REQUIRED
      IF(ITPE.EQ.ITPE1) GO TO 10
      ICOD=2
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C-------  FIND ELEMENT D.O.F.
10    CALL DLELM(KLOCE,VFG,VDIMP,VDLE)
C-------  COMPUTE AND PRINT STRESSES OR GRADIENTS
      ICOD=8
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
20    ITPE1=ITPE
      RETURN
      END
```

```
      SUBROUTINE ASRESD(IRESD,IREAC,KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,
     1  KNE,VKE,VFE,VKGS,VKGD,VKGI,VFG,VDLE,VRES,VREAC)
C=======================================================================
C     TO ASSEMBLE INTERNAL RESIDUALS IN VRES (IF IRESD .EQ.1)
C     AND EXTERNAL REACTIONS IN VREAC (IF IREAC.EQ.1)
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ELEM/NELT,NNEL,NTPE,NGRE,ME,NIDENT,MNULL
      COMMON/ASSE/NSYM,MFILLR(3)
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,IPG
     1  ,ICOD,NULL(3)
      COMMON/ES/M,MR,MP,M1,M2,MDUMMY(8)
      DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),VPRNE(*),VPREE(*),
     1  KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),VDLE(*),
     2  VRES(*),VREAC(*)
C-----------------------------------------------------------------------
C------- REWIND ELEMENT FILE (M2)
      REWIND M2
C------- LOOP OVER THE ELEMENTS
      DO 60 IE=1,NELT
C------- READ AN ELEMENT ON FILE M2
      CALL RDELEM(M2,KLOCE,VCORE,VPRNE,VPREE,KNE)
C------- EVALUATE INTERPOLATION FUNCTION IF REQUIRED
      IF(ITPE.EQ.ITPE1) GO TO 10
      ICOD=2
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C------- FIND ELEMENT D.O.F.
10    CALL DLELM(KLOCE,VFG,VDIMP,VDLE)
C------- EVALUATE ELEMENT REACTIONS
      ICOD=6
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C------- PRINT ELEMENT REACTIONS
      IF(M.GE.2) WRITE(MP,2000) IEL,(VFE(I),I=1,IDLE)
2000  FORMAT(/' REACTIONS (FE) , ELEMENT:',I5/(10X,10E12.5))
      IF(IRESD.NE.1) GO TO 20
C------- ASSEMBLE INTERNAL RESIDUALS
      CALL ASSEL(0,1,IDLE,NSYM,KLOCE,KLD,VKE,VFE,VKGS,VKGD,VKGI,VRES)
20    IF(IREAC.NE.1) GO TO 60
C------- ASSEMBLE EXTERNAL REACTIONS
C       MODIFY TERMS IN KLOCE SUCH THAT PRESCRIBED D.O.F. ARE THE ONLY
C       ASSEMBLED ONES
      DO 50 ID=1,IDLE
      IF(KLOCE(ID)) 30,50,40
30    KLOCE(ID)=-KLOCE(ID)
      GO TO 50
40    KLOCE(ID)=0
50    CONTINUE
      CALL ASSEL(0,1,IDLE,NSYM,KLOCE,KLD,VKE,VFE,VKGS,VKGD,VKGI,VREAC)
60    ITPE1=ITPE
      RETURN
      END
```

```
      SUBROUTINE ASSEL(IKG,IFG,IDLE,NSYM,KLOCE,KLD,VKE,VFE,VKGS,
     1  VKGD,VKGI,VFG)
C======================================================================
C     TO ASSEMBLE AN ELEMENT MATRIX AND/OR VECTOR
C     (MATRIX SYMMETRICAL OR NOT)
C       INPUT
C          IKG      IF IKG.EQ.1 ASSEMBLE ELEMENT MATRIX KE
C          IFG      IF IFG.EQ.1 ASSEMBLE ELEMENT VECTOR FE
C          IDLE     ELEMENT NUMBER OF D.O.F.
C          NSYM     0=SYMMETRIC PROBLEM, 1=UNSYMMETRIC PROBLEM
C          KLOCE    ELEMENT LOCALIZATION VECTOR
C          KLD      CUMULATIVE COLUMN HEIGHTS OF KG
C          VKE      ELEMENT MATRIX KE (FULL OR UPPER TRIANGLE BY
C                   DESCENDING COLUMNS)
C          VFE      ELEMENT VECTOR FE
C       OUTPUT
C          VKGS,VKGD,VKGI    GLOBAL MATRIX (SKYLINES)
C                   (SYMMETRIC OR NOT)
C          VFG      GLOBAL LOAD VECTOR
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION KLOCE(*),KLD(*),VKE(*),VFE(*),VKGS(*),VKGD(*),
     1  VKGI(*),VFG(*)
C----------------------------------------------------------------------
C
C-------  ASSEMBLE ELEMENT MATRIX
C
      IF(IKG.NE.1) GO TO 100
      IEQ0=IDLE
      IEQ1=1
C-------  FOR EACH COLUMN OF KE
      DO 90 JD=1,IDLE
      IF(NSYM.NE.1) IEQ0=JD
      JL=KLOCE(JD)
      IF(JL) 90,90,10
10    IO=KLD(JL+1)
      IEQ=IEQ1
      IQ=1
C-------  FOR EACH ROW OF KE
      DO 80 ID=1,IDLE
      IL=KLOCE(ID)
      IF(NSYM.EQ.1) GO TO 30
      IF(ID-JD) 30,20,20
20    IQ=ID
30    IF(IL) 80,80,40
40    IJ=JL-IL
      IF(IJ) 70,50,60
C-------  DIAGONAL TERMS OF KG
50    VKGD(IL)=VKGD(IL)+VKE(IEQ)
      GO TO 80
```

```
C-------  UPPER TRIANGLE TERMS OF KG
60    I=IO-IJ
      VKGS(I)=VKGS(I)+VKE(IEQ)
      GO TO 80
C-------  LOWER TRIANGLE TERMS OF KG
70    IF(NSYM.NE.1) GO TO 80
      I=KLD(IL+1)+IJ
      VKGI(I)=VKGI(I)+VKE(IEQ)
80    IEQ=IEQ+IQ
90    IEQ1=IEQ1+IEQ0
C
C-------  ASSEMBLE ELEMENT LOAD VECTOR
C
100   IF(IFG.NE.1) GO TO 130
      DO 120 ID=1,IDLE
      IL=KLOCE(ID)
      IF(IL) 120,120,110
110   VFG(IL)=VFG(IL)+VFE(ID)
120   CONTINUE
130   RETURN
      END
```

```
      SUBROUTINE MODFG(IDLE,NSYM,KLOCE,VDIMP,VKE,VFG)
C======================================================================
C     TO MODIFY VECTOR FG TO TAKE INTO ACCOUNT OF PRESCRIBED NON ZERO
C     D.O.F. FOR A GIVEN ELEMENT
C       INPUT
C          IDLE    ELEMENT NUMBER OF D.O.F.
C          NSYM    0=SYMMETRIC PROBLEM, 1=NON SYMMETRIC PROBLEM
C          KLOCE   ELEMENT LOCALIZATION VECTOR
C          VDIMP   VALUES OF PRESCRIBED D.O.F.
C          VKE     ELEMENT MATRIX (FULL OR UPPER TRIANGLE
C                  BY DESCENDING COLUMNS)
C       OUTPUT
C          VFG     GLOBAL LOAD VECTOR
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION KLOCE(*),VDIMP(*),VKE(*),VFG(*)
      DATA ZERO/0.D0/
C----------------------------------------------------------------------
      IEQ0=IDLE
      IEQ1=1
C------- FOR EACH ROW OF ELEMENT MATRIX
      DO 50 JD=1,IDLE
      IF(NSYM.NE.1) IEQ0=JD
      IEQ=IEQ1
      JL=KLOCE(JD)
      IQ=1
      IF(JL) 10,50,50
10    JL=-JL
      DIMP=VDIMP(JL)
      IF(DIMP.EQ.ZERO) GO TO 50
C------- FOR EACH COLUMN OF ELEMENT MATRIX
      DO 40 ID=1,IDLE
      IL=KLOCE(ID)
      IF(NSYM.EQ.1) GO TO 30
      IF(ID-JD) 30,20,20
20    IQ=ID
30    IF(IL.GT.0) VFG(IL)=VFG(IL)-VKE(IEQ)*DIMP
40    IEQ=IEQ+IQ
50    IEQ1=IEQ1+IEQ0
      RETURN
      END
```

```
      SUBROUTINE PRPVTS(VKGD)
C=====================================================================
C   TO EVALUATE AND TO PRINT THE PIVOTS AND THE DETERMINANT OF MATRIX KG
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/ES/M,MR,MP,MDUMMY(10)
      DIMENSION VKGD(*)
      DATA UN/1.D0/,GROS/1.D38/
      ABS(X)=DABS(X)
C---------------------------------------------------------------------
      X1=GROS
      X2=GROS
      DET=UN
      IDET=0
C------- PRINT PIVOTS OF MATRIX KG
      IF(M.GE.2) WRITE(MP,2000)(VKGD(I),I=1,NEQ)
2000  FORMAT(/' GLOBAL MATRIX PIVOTS'/(1X,10E12.5))
      DO 50 I=1,NEQ
C------- ABSOLUTE VALUE OF MINIMUM PIVOT
      X=ABS(VKGD(I))
      IF(X.GT.X1) GO TO 10
      X1=X
      I1=I
C------- ALGEBRAIC VALUE OF MINIMUM PIVOT
      X=VKGD(I)
10    IF(X.GT.X2) GO TO 20
      X2=X
      I2=I
C------- DETERMINANT (BOUNDS : 10 EXPONENT + OR - 10)
20    DET=DET*VKGD(I)
30    DET1=ABS(DET)
      IF(DET1.LT.1.D10) GO TO 40
      DET=DET*1.D-10
      IDET=IDET+10
40    IF(DET1.GT.1.D-10) GO TO 50
      DET=DET*1.D10
      IDET=IDET-10
      GO TO 30
50    CONTINUE
C------- OUTPUT
      WRITE(MP,2010) X1,I1,X2,I2,DET,IDET
2010  FORMAT(/15X,'ABSOLUTE VALUE OF MINIMUM PIVOT   =',E12.5,' EQUATION
     1:',I5   /29X,               'ALGEBRAIC VALUE=',E12.5,' EQUATION:',
     2  I5   /29X,               'DETERMINANT       =',E12.5,' * 10 ** ',
     3  I5/)
      RETURN
      END
```

```
      SUBROUTINE PRRESD(VKGS,VKGD,VKGI,VFG,KLD,VRES)
C=================================================================
C     TO COMPUTE AND PRINT THE RESIDUAL VECTOR K.U - F
C=================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ASSE/NSYM,NKG,MFILLR(2)
      COMMON/RESO/NEQ,NRES,MRES
      COMMON/ES/M,MR,MP,M1,M2,M3,MDUMMY(7)
      DIMENSION VKGS(*),VKGD(*),VKGI(*),VFG(*),KLD(*),VRES(*)
      DATA ZERO/0.D0/
      ABS(X)=DABS(X)
C-----------------------------------------------------------------
      REWIND M3
C------- SKIP VECTOR FG NON MODIFIED BY B.C. ON FILE M3
      READ(M3) (VRES(I),I=1,NEQ)
C------- READ VECTOR FG MODIFIED BY B.C. AND MATRIX KG
      READ(M3) (VRES(I),I=1,NEQ)
      READ(M3) (VKGS(I),I=1,NKG),(VKGD(I),I=1,NEQ)
      IF(NSYM.EQ.1) READ(M3) (VKGI(I),I=1,NKG)
C------- EVALUATE THE RESIDUAL VECTOR
      DO 10 I=1,NEQ
10    VRES(I)=-VRES(I)
      CALL MULKU(VKGS,VKGD,VKGI,KLD,VFG,NEQ,NSYM,VRES)
      DO 20 I=1,NEQ
20    VRES(I)=-VRES(I)
      X1=ZERO
      DO 30 I=1,NEQ
      X=ABS(VRES(I))
      IF(X1.GE.X) GO TO 30
      X1=X
      I1=I
30    CONTINUE
      IF(M.GE.2) WRITE(MP,2000) (VRES(I),I=1,NEQ)
2000  FORMAT(/' RESIDUALS VECTOR'/(1X,10E12.5))
      WRITE(MP,2010) X1,I1
2010  FORMAT(/' MAX. RESIDUAL VALUE=',E12.5,' EQUATION',I5)
      RETURN
      END
```

```
      SUBROUTINE PRSOL(KDLNC,VCORG,VDIMP,KNEQ,VFG)
C=====================================================================
C     TO PRINT THE SOLUTION
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 RF,RL,FX
      COMMON/COOR/NDIM,NNT,NNULL(2),FNULL(3)
      COMMON/ES/M,MR,MP,MDUMMY(10)
      COMMON/TRVL/V(10),FX(10),RDUMMY(506),NULL
      DIMENSION VDIMP(*),KDLNC(*),VCORG(*),KNEQ(*),VFG(*)
      DATA RF/' *  '/,RL/'    '/,ZERO/0.D0/
C---------------------------------------------------------------------
      X2=ZERO
      X3=ZERO
      WRITE(MP,2000)
2000  FORMAT(/' NODES',4X,'X',11X,'Y',11X,'Z',10X,'DEGREES OF FREEDOM (*
     1 = PRESCRIBED)'/)
      I2=0
      DO 50 IN=1,NNT
      I1=I2+1
      I2=I2+NDIM
      ID1=KDLNC(IN)+1
      ID2=KDLNC(IN+1)
      ID=ID2-ID1+1
      IF(ID2.LT.ID1) GO TO 50
      X1=VCORG(I1)
      IF(NDIM.GE.2) X2=VCORG(I1+1)
      IF(NDIM.GE.3) X3=VCORG(I1+2)
      J=ID1
      DO 40 I=1,ID
      JJ=KNEQ(J)
      IF(JJ) 10,20,30
10    V(I)=VDIMP(-JJ)
      FX(I)=RF
      GO TO 40
20    V(I)=ZERO
      FX(I)=RF
      GO TO 40
30    V(I)=VFG(JJ)
      FX(I)=RL
40    J=J+1
      WRITE(MP,2010)IN,X1,X2,X3,(V(II),FX(II),II=1,ID)
2010  FORMAT(1X,I5,3E12.5,5X,5(E12.5,A4)/47X,5(E12.5,A4))
50    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DLELM(KLOCE,VDLG,VDIMP,VDLE)
C=================================================================
C     TO GENERATE ELEMENT D.O.F.
C=================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/RGDT/IEL,INUL(3),IDLE,NULL(10)
      COMMON/ES/M,MR,MP,MDUMMY(10)
      DIMENSION KLOCE(*),VDLG(*),VDIMP(*),VDLE(*)
      DATA ZERO/0.D0/
C-----------------------------------------------------------------
      DO 40 ID=1,IDLE
      IL=KLOCE(ID)
      IF(IL) 10,20,30
10    VDLE(ID)=VDIMP(-IL)
      GO TO 40
20    VDLE(ID)=ZERO
      GO TO 40
30    VDLE(ID)=VDLG(IL)
40    CONTINUE
      IF(M.GE.2) WRITE(MP,2000) IEL,(VDLE(ID),ID=1,IDLE)
2000  FORMAT(' DEGREES OF FREEDOM OF ELEMENT ',I5/(1X,10E12.5))
      RETURN
      END
```

```
      SUBROUTINE MULKU(VKGS,VKGD,VKGI,KLD,VFG,NEQ,NSYM,VRES)
C=======================================================================
C     SUBPROGRAM :
C     TO ADD VECTOR RES TO THE PRODUCT OF MATRIX KG AND THE VECTOR FG
C       INPUT
C          VKGS,VKGD,VKGI   MATRIX KG STORED BY SKYLINE
C                           (SYM. OR NON SYM.)
C          KLD     ARRAY OF ADDRESS OF COLUMN TOP TERMS IN KG
C          VFG     VECTOR FG
C       .  NEQ     ORDER OF VECTORS FG AND RES
C          NSYM    .EQ.1 IF NON SYMMETRIC PROBLEM
C          VRES    VECTOR RES
C       OUTPUT
C          VRES    VECTOR RES
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION VKGS(*),VKGD(*),VKGI(*),KLD(*),VFG(*),VRES(*)
C-----------------------------------------------------------------------
C------- FOR EACH COLUMN OF MATRIX KG
      DO 20 IK=1,NEQ
      JHK=KLD(IK)
      JHK1=KLD(IK+1)
      LHK=JHK1-JHK
C------- DIAGONAL TERMS
      C=VKGD(IK)*VFG(IK)
      IF(LHK.LE.0) GO TO 20
      I0=IK-LHK
C------- ROW TERMS
      IF(NSYM.NE.1) C=C+SCAL(VKGS(JHK),VFG(I0),LHK)
      IF(NSYM.EQ.1) C=C+SCAL(VKGI(JHK),VFG(I0),LHK)
C------- COLUMN TERMS
      J=JHK
      I1=IK-1
      DO 10 IJ=I0,I1
      VRES(IJ)=VRES(IJ)+VKGS(J)*VFG(IK)
10    J=J+1
20    VRES(IK)=VRES(IK)+C
      RETURN
      END
```

```
      SUBROUTINE GAUSS(IPGKED,NDIM,VKPG,VCPG,IPG)
C=====================================================================
C     TO FORM ARRAYS OF COORDINATES AND WEIGHTS AT GAUSS POINTS
C     (1,2 AND 3 DIMENSIONS)(1,2,3 OR 4 G.P. PER DIMENSION)
C       INPUT
C          IPGKED  NUMBER OF POINTS IN KSI,ETA,ZETA DIRECTIONS
C          NDIM    NUMBER OF DIMENSIONS (1,2 OR 3)
C       OUTPUT
C          VKPG    COORDINATES OF GAUSS POINTS
C          VCPG    WEIGHTS AT GAUSS POINTS
C          IPG     TOTAL NUMBER OF GAUSS POINTS
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION IPGKED(*),VKPG(*),VCPG(*),G(10),P(10),INDIC(4)
C
C+++       THIS IS COMMENTED OUT BECAUSE OF THE MS FORTRAN COMP-
C+++       ILER BUG WHICH WILL NOT INITIALIZE $LARGE ARRAYS.
C+++       THESE ARRAYS ARE NOW INITIALIZED BY A CALL TO A DUMMY
C+++       SUBROUTINE INITGA WHICH EXISTS SOLELY TO INITIALIZE
C+++       THESE THREE ARRAYS
C
C     DATA INDIC/1,2,4,7/
C     DATA G/0.0D0,-.577350269189626D0,.577350269189626D0,
C    1       -.774596669241483D0,0.0D0,.774596669241483D0,
C    2       -.861136311594050D0,-.339981043584860D0,
C    3       .339981043584860D0,.861136311594050D0/
C     DATA P/2.0D0,1.0D0,1.0D0,
C    1      0.555555555555556D0,0.888888888888889D0,0.555555555555556D0,
C    2       .347854845137450D0,.652145154862550D0,
C    3       .652145154862550D0,.347854845137450D0/
C
C         HERE IS THE CALL TO GET AROUND THE MICROSOFT
C         COMPILER BUG
C
      CALL INITGA(INDIC,G,P)
C
C+++       ALL OF THIS HAS BEEN TO GET AROUND THE MICROSOFT
C+++       COMPILER BUG
C
C---------------------------------------------------------------------
      II=IPGKED(1)
      IMIN=INDIC(II)
      IMAX=IMIN+II-1
      IF(NDIM-2) 10,20,30
C------- 1 DIMENSION
10    IPG=0
      DO 15 I=IMIN,IMAX
      IPG=IPG+1
      VKPG(IPG)=G(I)
15    VCPG(IPG)=P(I)
```

```
      RETURN
C-------  2 DIMENSIONS
20    II=IPGKED(2)
      JMIN=INDIC(II)
      JMAX=JMIN+II-1
      IPG=0
      L=1
      DO 25 I=IMIN,IMAX
      DO 25 J=JMIN,JMAX
      IPG=IPG+1
      VKPG(L)=G(I)
      VKPG(L+1)=G(J)
      L=L+2
25    VCPG(IPG)=P(I)*P(J)
      RETURN
C-------  3 DIMENSIONS
30    II=IPGKED(2)
      JMIN=INDIC(II)
      JMAX=JMIN+II-1
      II=IPGKED(3)
      KMIN=INDIC(II)
      KMAX=KMIN+II-1
      IPG=0
      L=1
      DO 35 I=IMIN,IMAX
      DO 35 J=JMIN,JMAX
      DO 35 K=KMIN,KMAX
      IPG=IPG+1
      VKPG(L)=G(I)
      VKPG(L+1)=G(J)
      VKPG(L+2)=G(K)
      L=L+3
35    VCPG(IPG)=P(I)*P(J)*P(K)
      RETURN
      END
```

```
      SUBROUTINE PNINV(VKSI,KEXP,VP,K1,VPN)
C=====================================================================
C     EVALUATE  THE PN-INVERSE MATRIX WHICH
C     CONTAINS THE COEFFICIENTS OF FUNCTIONS N
C       INPUT        VKSI,KEXP,INEL,IDLE,ITPE,M,MP
C       WORKSPACE    VP,K1
C       OUTPUT       VPN
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNULL(3),FNULL(3)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,IPG
     1  ,NULL(4)
      COMMON/ES/M,MR,MP,MDUMMY(10)
      DIMENSION VKSI(*),KEXP(*),VP(*),K1(*),VPN(*),KDER(3)
      DATA ZERO/0.D0/
C
C+++     THIS IS TO GET AROUND THE MICROSOFT COMPILER BUG
C+++     WHICH WILL NOT INITIALIZE $LARGE ARRAYS
C     DATA KDER/3*0/
C
      KDER(1) = 0
      KDER(2) = 0
      KDER(3) = 0
C+++     ALL THIS HAS BEEN TO GET AROUND THE MICROSOFT
C+++     COMPILER BUG
C
C---------------------------------------------------------------------
C
C.......  FORM PN MATRIX (FOR ANY LAGRANGE TYPE ELEMENT)
C
C
      I0=1
      I1=1
      DO 20 IN=1,INEL
      CALL BASEP(VKSI(I1),KEXP,KDER,VP)
      I2=I0
      DO 10 IJ=1,INEL
      VPN(I2)=VP(IJ)
10    I2=I2+INEL
      I0=I0+1
20    I1=I1+NDIM
C
C.......  END OF PN FORMATION
C
C-------  PRINT THE PN MATRIX
      IF(M.LT.4) GO TO 40
      WRITE(MP,2000)
2000  FORMAT(/' PN MATRIX'/)
      I0=(INEL-1)*INEL
      DO 30 IO=1,INEL
```

```
      I1=I0+ID
30    WRITE(MP,2010) (VPN(IJ),IJ=I0,I1,INEL)
2010  FORMAT(1X,10E13.5/(14X,9E13.5))
C------  INVERSE THE PN MATRIX
40    CALL INVERS(VPN,INEL,INEL,K1,DET)
      IF(DET.NE.ZERO) GO TO 50
      WRITE(MP,2020) ITPE
2020  FORMAT(' *** ERROR, PN SINGULAR, ELEMENT TYPE:',I3)
      STOP
C------  PRINT THE PN-INVERSE MATRIX
50    IF(M.LT.4) GO TO 70
      WRITE(MP,2030)
2030  FORMAT(/' PN-INVERSE MATRIX'/)
      DO 60 I0=1,INEL
      I1=I0+ID
60    WRITE(MP,2010) (VPN(IJ),IJ=I0,I1,INEL)
70    RETURN
      END
```

```
$LARGE
$DEBUG
$NOFLOATCALLS
$D066
      SUBROUTINE NI(VKSI,KEXP,KDER,VP,VPN,VNI)
C=====================================================================
C     TO EVALUATE FUNCTIONS N OR THEIR DERIVATIVES
C     AT POINT VKSI ON THE REFERENCE ELEMENT
C       INPUT    VKSI,KEXP,KDER,VP,VPN,IDLE,M,MP
C       OUTPUT   VNI
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNULL(3),FNULL(3)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,IPG
     1  ,NULL(4)
      COMMON/ES/M,MR,MP,MDUMMY(10)
      DIMENSION VKSI(*),KEXP(*),KDER(*),VP(*),VPN(*),VNI(*)
      DATA ZERO/0.D0/
C---------------------------------------------------------------------
C-------  COMPUTE THE POLYNOMIAL BASIS AT POINT VKSI
      CALL BASEP(VKSI,KEXP,KDER,VP)
C-------  P*(PN-INVERSE) PRODUCT
      I0=1
      DO 20 IJ=1,INEL
      I1=I0
      C=ZERO
      DO 10 II=1,INEL
      C=C+VP(II)*VPN(I1)
10    I1=I1+1
      VNI(IJ)=C
20    I0=I0+INEL
C-------  PRINT FUNCTIONS N
      IF(M.LT.3) GO TO 30
      WRITE(MP,2000) (KDER(I),I=1,NDIM)
2000  FORMAT(/' DERIVATIVE OF N WITH ORDER ',3I2)
      WRITE(MP,2010) (VKSI(I),I=1,NDIM)
2010  FORMAT(14X,'AT POINT ',3E13.5)
      WRITE(MP,2020) (VNI(I),I=1,INEL)
2020  FORMAT(/(1X,10E13.5))
30    RETURN
      END
```

```
      SUBROUTINE BASEP(VKSI,KEXP,KDER,VP)
C=======================================================================
C     TO EVALUATE THE POLYNOMIAL BASIS AND ITS DERIVATIVES AT POINT VKSI
C       INPUT    VKSI,KEXP,KDER,IDLE,IDEG,NDIM,M,MP
C       OUTPUT   VP
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNULL(3),FNULL(3)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,IPG
     1  ,NULL(4)
      COMMON/ES/M,MR,MP,MDUMMY(10)
      DIMENSION VKSI(*),KEXP(*),KDER(*),VP(*)
      DIMENSION PUISS(3,10)
      DATA ZERO/0.D0/,UN/1.D0/
C-----------------------------------------------------------------------
C------- FORM SUCCESSIVE POWERS OF KSI,ETA,DZETA
      DO 10 I=1,NDIM
      PUISS(I,1)=UN
      DO 10 ID=1,IDEG
10    PUISS(I,ID+1)=PUISS(I,ID)*VKSI(I)
C------- DERIVATIVES OF ORDER KDER WITH RESPECT TO KSI,ETA,DZETA
      DO 50 IDL=1,INEL
      C1=UN
      I0=(IDL-1)*NDIM
      DO 30 I=1,NDIM
      IDR=KDER(I)
      I0=I0+1
      IXP=KEXP(I0)+1
      J=IXP-IDR
      IF(J.LE.0) GO TO 40
      IF(IDR.LE.0) GO TO 30
      DO 20 ID=1,IDR
20    C1=C1*(IXP-ID)
30    C1=C1*PUISS(I,J)
      GO TO 50
40    C1=ZERO
50    VP(IDL)=C1
C------- PRINT POLYNOMIAL BASIS
      IF(M.LT.4) GO TO 60
      WRITE(MP,2000) (KDER(I),I=1,NDIM)
2000  FORMAT(/' POLYNOMIAL BASIS, DERIVATIVE OF ORDER ',3I2)
      WRITE(MP,2010) (VKSI(I),I=1,NDIM)
2010  FORMAT(19X,'AT POINT ',3E13.5)
      WRITE(MP,2020) (VP(I),I=1,INEL)
2020  FORMAT(/(1X,10E12.5))
60    RETURN
      END
```

```
      SUBROUTINE INVERS(VP,N,IVP,K,DET)
C===================================================================
C     TO INVERT A NON-SYMMETRIC MATRIX WITH SEARCH OF A
C     NON-ZERO PIVOT IN A COLUMN
C       INPUT
C          VP      MATRIX TO BE INVERTED
C          N       ORDER OF THE MATRIX
C          IVP     DIMENSION OF THE MATRIX IN THE CALLING PROGRAM
C          K       INTEGER WORKING ARRAY WITH LENGTH N
C       OUTPUT
C          VP      INVERSE MATRIX
C          DET     DETERMINANT
C===================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION VP(IVP,IVP),K(N)
      DATA ZERO/0.D0/,UN/1.D0/,EPS/1.D-13/
      ABS(X)=DABS(X)
C-------------------------------------------------------------------
      DET=UN
      DO 5 I=1,N
5     K(I)=I
C------- START INVERSION
      DO 80 II=1,N
C------- SEARCH FOR NON-ZERO PIVOT IN COLUMN II
      DO 10 I=II,N
      PIV=VP(I,II)
      IF(ABS(PIV).GT.EPS) GO TO 20
10    CONTINUE
      DET=ZERO
      RETURN
C------- EXCHANGE LINES II AND I
20    DET=DET*PIV
      IF(I.EQ.II) GO TO 40
      I1=K(II)
      K(II)=K(I)
      K(I)=I1
      DO 30 J=1,N
      C=VP(I,J)
      VP(I,J)=VP(II,J)
30    VP(II,J)=C
      DET=-DET
C------- NORMALIZE PIVOT LINE
40    C=UN/PIV
      VP(II,II)=UN
      DO 50 J=1,N
50    VP(II,J)=VP(II,J)*C
C------- ELIMINATION
      DO 70 I=1,N
      IF(I.EQ.II) GO TO 70
      C=VP(I,II)
```

```
      VP(I,II)=ZERO
      DO 60 J=1,N
60    VP(I,J)=VP(I,J)-C*VP(II,J)
70    CONTINUE
80    CONTINUE
C-------  REORDER THE COLUMNS OF INVERSE MATRIX
      DO 120 J=1,N
C-------  FIND J1 SUCH THAT K(J1)=J
      DO 90 J1=J,N
      JJ=K(J1)
      IF(JJ.EQ.J) GO TO 100
90    CONTINUE
100   IF(J.EQ.J1) GO TO 120
C-------  EXCHANGE COLUMNS J AND J1
      K(J1)=K(J)
      DO 110 I=1,N
      C=VP(I,J)
      VP(I,J)=VP(I,J1)
110   VP(I,J1)=C
120   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE JACOB(VNI,VCORE,NDIM,INEL,VJ,VJ1,DETJ)
C=================================================================
C     TO EVALUATE THE JACOBIAN MATRIX, ITS DETERMINANT AND
C     ITS INVERSE (1,2,3 DIMENSIONS)
C       INPUT
C          VNI     DERIVATIVES OF INTERPOLATION FUNCTION W.R.T.
C                  KSI,ETA,DZETA
C          VCORE   ELEMENT NODAL COORDINATES
C          NDIM    NUMBER OF DIMENSIONS
C          INEL    NUMBER OF NODES PER ELEMENT
C       OUTPUT
C          VJ      JACOBIAN MATRIX
C          VJ1     INVERSE OF JACOBIAN MATRIX
C          DETJ    DETERMINANT OF JACOBIAN MATRIX
C=================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION VNI(INEL,*),VCORE(NDIM,*),VJ(*),VJ1(*)
      DATA ZERO/0.D0/,UN/1.D0/
C-----------------------------------------------------------------
C------- FORM THE JACOBIAN MATRIX
      J=1
      DO 20 JJ=1,NDIM
      DO 20 II=1,NDIM
      C=ZERO
      DO 10 IJ=1,INEL
10    C=C+VNI(IJ,II)*VCORE(JJ,IJ)
      VJ(J)=C
20    J=J+1
C------ 1, 2, OR 3 DIMENSIONAL INVERSION
      GO TO (40,50,60),NDIM
40    DETJ=VJ(1)
      IF(DETJ.EQ.ZERO) RETURN
      VJ1(1)=UN/DETJ
      RETURN
50    DETJ=VJ(1)*VJ(4)-VJ(2)*VJ(3)
      IF(DETJ.EQ.ZERO) RETURN
      VJ1(1)=VJ(4)/DETJ
      VJ1(2)=-VJ(2)/DETJ
      VJ1(3)=-VJ(3)/DETJ
      VJ1(4)=VJ(1)/DETJ
      RETURN
60    DETJ=VJ(1)*(VJ(5)*VJ(9)-VJ(8)*VJ(6))
     1    +VJ(4)*(VJ(8)*VJ(3)-VJ(2)*VJ(9))
     2    +VJ(7)*(VJ(2)*VJ(6)-VJ(5)*VJ(3))
      IF(DETJ.EQ.ZERO) RETURN
      VJ1(1)=(VJ(5)*VJ(9)-VJ(6)*VJ(8))/DETJ
      VJ1(2)=(VJ(3)*VJ(8)-VJ(2)*VJ(9))/DETJ
      VJ1(3)=(VJ(2)*VJ(6)-VJ(3)*VJ(5))/DETJ
      VJ1(4)=(VJ(7)*VJ(6)-VJ(4)*VJ(9))/DETJ
      VJ1(5)=(VJ(1)*VJ(9)-VJ(7)*VJ(3))/DETJ
```

```
      VJ1(6)=(VJ(4)*VJ(3)-VJ(6)*VJ(1))/DETJ
      VJ1(7)=(VJ(4)*VJ(8)-VJ(7)*VJ(5))/DETJ
      VJ1(8)=(VJ(2)*VJ(7)-VJ(8)*VJ(1))/DETJ
      VJ1(9)=(VJ(1)*VJ(5)-VJ(4)*VJ(2))/DETJ
      RETURN
      END


      SUBROUTINE DNIDX(VNI,VJ1,NDIM,INEL,VNIX)
C======================================================================
C     COMPUTE  THE DERIVATIVES OF INTERPOLATION FUNCTIONS WITH
C     RESPECT TO X,Y,Z
C     (1,2 OR 3 DIMENSIONS)
C       INPUT
C          VNI     DERIVATIVES OF INTERPOLATION FUNCTIONS WITH RESPECT
C                  TO  KSI,ETA,DZETA
C          VJ1     INVERSE OF THE JACOBIAN
C          NDIM    NUMBER OF DIMENSIONS (1,2 OR 3)
C          INEL    NUMBER OF INTERPOLATION FUNCTIONS (OR NODES)
C       OUTPUT
C          VNIX    X,Y,Z DERIVATIVES OF INTERPOLATION FUNCTIONS
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION VNI(INEL,*),VJ1(NDIM,*),VNIX(INEL,*)
      DATA ZERO/0.D0/
C----------------------------------------------------------------------
      DO 20 I=1,NDIM
      DO 20 J=1,INEL
      C=ZERO
      DO 10 IJ=1,NDIM
10    C=C+VJ1(I,IJ)*VNI(J,IJ)
20    VNIX(J,I)=C
      RETURN
      END
```

```
      SUBROUTINE SOL(VKGS,VKGD,VKGI,VFG,KLD,NEQ,MP,IFAC,ISOL,NSYM,ENERG)
C=======================================================================
C     TO SOLVE A LINEAR SYSTEM (SYMMETRICAL OR NOT).
C     THE MATRIX IS STORED IN CORE BY SKYLINES  IN ARRAYS
C     VKGS,VKGD,VKGI
C       INPUT
C          VKGS,VKGD,VKGI    SYSTEM MATRIX : UPPER, DIAGONAL AND
C                            LOWER PARTS
C          VFG               SECOND MEMBER
C          KLD               ADDRESSES OF COLUMN TOP TERMS
C          NEQ               NUMBER OF EQUATIONS
C          MP                OUTPUT DEVICE NUMBER
C          IFAC              IF IFAC.EQ.1 TRIANGULARIZE THE
C                            MATRIX
C          ISOL              IF ISOL.EQ.1 COMPUTE THE SOLUTION FROM
C                            TRIANGULARIZED MATRIX
C          NSYM              INDEX FOR NONSYMMETRIC PROBLEM
C       OUTPUT
C          VKGS,VKGD,VKGI    TRIANGULARIZED MATRIX (IF IFAC.EQ.1)
C          VFG               SOLUTION (IF ISOL.EQ.1)
C          ENERG             SYSTEM ENERGY (IF NSYM.EQ.0)
C=======================================================================
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION VKGS(*),VKGD(*),VKGI(*),VFG(*),KLD(*)
      DATA ZERO/0.0D0/
C-----------------------------------------------------------------------
      IK=1
      IF(VKGD(1).EQ.ZERO) GO TO 80
      ENERG=ZERO
C
C----- FOR EACH COLUMN IK TO BE MODIFIED .
C
      JHK=1
      DO 100 IK=2,NEQ
C----- ADDRESS OF THE NEXT COLUMN TOP TERM IK+1
      JHK1=KLD(IK+1)
C----- HEIGHT OF COLUMN IK (INCLUDE UPPER AND DIAGONAL TERMS)
      LHK=JHK1-JHK
      LHK1=LHK-1
C----- ROW OF FIRST TERM TO BE MODIFIED IN COLUMN IK
      IMIN=IK-LHK1
      IMIN1=IMIN-1
C------- ROW OF LAST TERM TO BE MODIFIED IN COLUMN IK
      IMAX=IK-1
      IF(LHK1.LT.0) GO TO 100
      IF(IFAC.NE.1) GO TO 90
      IF(NSYM.EQ.1) VKGI(JHK)=VKGI(JHK)/VKGD(IMIN1)
      IF(LHK1.EQ.0) GO TO 40
C
C----- MODIFY NON-DIAGONAL TERM IN COLUMN IK
```

```
C
      JCK=JHK+1
      JHJ=KLD(IMIN)
C----  FOR EACH TERM LOCATED AT JCK AND CORRESPONDING TO COLUMN IJ
      DO 30 IJ=IMIN,IMAX
      JHJ1=KLD(IJ+1)
C----- NUMBER OF MODIFICATIVE TERMS FOR COEFFICIENT LOCATED AT JCK
      IC=MIN0(JCK-JHK,JHJ1-JHJ)
      IF(IC.LE.0.AND.NSYM.EQ.0) GO TO 20
      C1=ZERO
      IF(IC.LE.0) GO TO 17
      J1=JHJ1-IC
      J2=JCK-IC
      IF(NSYM.EQ.1) GO TO 15
      VKGS(JCK)=VKGS(JCK)-SCAL(VKGS(J1),VKGS(J2),IC)
      GO TO 20
15    VKGS(JCK)=VKGS(JCK)-SCAL(VKGI(J1),VKGS(J2),IC)
      C1=SCAL(VKGS(J1),VKGI(J2),IC)
17    VKGI(JCK)=(VKGI(JCK)-C1)/VKGD(IJ)
20    JCK=JCK+1
30    JHJ=JHJ1
C
C----- MODIFY DIAGONAL TERM
C
40    JCK=JHK
      CDIAG=ZERO
      DO 70 IJ=IMIN1,IMAX
      C1=VKGS(JCK)
      IF(NSYM.EQ.1) GO TO 50
      C2=C1/VKGD(IJ)
      VKGS(JCK)=C2
      GO TO 60
50    C2=VKGI(JCK)
60    CDIAG=CDIAG+C1*C2
70    JCK=JCK+1
      VKGD(IK)=VKGD(IK)-CDIAG
      IF(VKGD(IK)) 90,80,90
80    WRITE(MP,2000) IK
2000  FORMAT(' *** ERROR,ZERO PIVOT EQUATION ',I5)
      STOP
C
C----- SOLVE LOWER TRIANGULAR SYSTEM
C
   90 IF(ISOL.NE.1) GO TO 100
      IF(NSYM.NE.1) VFG(IK)=VFG(IK)-SCAL(VKGS(JHK),VFG(IMIN1),LHK)
      IF(NSYM.EQ.1) VFG(IK)=VFG(IK)-SCAL(VKGI(JHK),VFG(IMIN1),LHK)
100   JHK=JHK1
      IF(ISOL.NE.1) RETURN
C
C----- SOLVE DIAGONAL SYSTEM
C
```

```
      IF(NSYM.EQ.1) GO TO 120
      DO 110 IK=1,NEQ
      C1=VKGD(IK)
      C2=VFG(IK)/C1
      VFG(IK)=C2
110   ENERG=ENERG+C1*C2*C2
C
C-----  SOLVE DIAGONAL SYSTEM
C
120   IK=NEQ+1
      JHK1=KLD(IK)
130   IK=IK-1
      IF(NSYM.EQ.1) VFG(IK)=VFG(IK)/VKGD(IK)
      IF(IK.EQ.1) RETURN
      C1=VFG(IK)
      JHK=KLD(IK)
      JBK=JHK1-1
      IF(JHK.GT.JBK)GO TO 150
      IJ=IK-JBK+JHK-1
      DO 140 JCK=JHK,JBK
      VFG(IJ)=VFG(IJ)-VKGS(JCK)*C1
140   IJ=IJ+1
150   JHK1=JHK
      GO TO 130
      END
      FUNCTION SCAL(X,Y,N)
C=================================================================
C     INNER PRODUCT OF VECTORS X AND Y OF LENGTH N
C       (FUNCTION TO BE WRITTEN EVENTUALLY IN ASSEMBLER)
C=================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(*),Y(*)
      DATA ZERO/0.0D0/
C-----------------------------------------------------------------
      SCAL=ZERO
      DO 10 I=1,N
10    SCAL=SCAL+X(I)*Y(I)
      RETURN
      END
```

```
      SUBROUTINE BLLIND
C=====================================================================
C     TO CALL BLOCK 'LIND'
C     TO ASSEMBLE AND TO SOLVE A LINEAR PROBLEM WHEN MATRIX KG IS
C     STORED BLOCKWISE ON DISK
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/COOR/NDIM,NNT,NDLN,NDLT,FNULL(3)
      COMMON/ELEM/NULL(4),ME,MNULL(2)
      COMMON/ASSE/NSYM,NKG,NKE,NDLE
      COMMON/RESO/NEQ,NRES,MRES
      COMMON/LIND/NLBL,NBLM,MKG1,MKG2
      COMMON/ES/M,MR,MP,M1,M2,M3,M4,M5,MDUMMY(5)
      COMMON/ALLOC/NVA,IVA,IVAMAX,NREEL,IDUMMY
      COMMON/LOC/LCORG,LDLNC,LNEQ,LDIMP,LPRNG,LPREG,LLD,LLOCE,LCORE,LNE,
     1  LPRNE,LPREE,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,LRES,LDLG,LDUMMY(4)
      COMMON VA(1)
      DIMENSION TBL(10),IN(3)
      DATA DEUX/2.D0/,NBLMAX/100/
C+++     THIS IS COMMENTED OUT BECAUSE OF AN MS FORTRAN COMPILER
C+++     BUG WHICH WILL NOT INITIALIZE  $LARGE ARRAYS.  THIS ARRAY
C+++     IS NOW INITIALIZED BY A CALL TO A DUMMY SUBROUTINE
C+++     INITBL WHICH EXISTS SOLELY TO INITIALIZE THIS ARRAY
C
C     DATA TBL/'KGS ','KGD ','KGI ','FG  ','KE  ','FE  ','RES ','DLE ',
C    1 'EB  ','PB  '/
C
C        HERE IS THE CALL TO GET AROUND THE COMPILER BUG
      CALL INITBL(TBL,'LIND')
C
C+++     ALL OF THIS IS TO GET AROUND THE MICROSOFT
C+++     COMPILER BUG
C
C---------------------------------------------------------------------
C------- FILE NUMBERS
      IF(M1.EQ.0) M1=MR
      IF(M2.EQ.0) M2=ME
      IF(M3.EQ.0) M3=MRES
      IF(M4.EQ.0) M4=MKG1
      IF(M5.EQ.0) M5=MKG2
      OPEN(M3,FILE='$$M3$.DAT',STATUS='NEW',FORM='UNFORMATTED')
      OPEN(M4,FILE='$$M4$.DAT',STATUS='NEW',FORM='UNFORMATTED')
      OPEN(M5,FILE='$$M5$.DAT',STATUS='NEW',FORM='UNFORMATTED')
C------- READ BLOCK PARAMETERS
      READ(M1,1000) IN
1000  FORMAT(3I5)
      IF(IN(1).NE.0) NRES=1
      NLBL=IN(2)
      NBLM=IN(3)
```

```
      WRITE(MP,2000) M,NRES
2000  FORMAT(//' ON DISK ASSEMBLAGE AND LINEAR SOLUTION (M=',I2,')'/
     1  ' ',42('=')/15X,'INDEX FOR RESIDUAL COMPUTATION    (NRES)=',I5)
      IF(LKGD.EQ.1) CALL ESPACE(NEQ,1,TBL(2),LKGD)
      IF(LFG.EQ.1) CALL ESPACE(NEQ,1,TBL(4),LFG)
      IF(LKE.EQ.1) CALL ESPACE(NKE,1,TBL(5),LKE)
      IF(LFE.EQ.1) CALL ESPACE(NDLE,1,TBL(6),LFE)
      IF(LRES.EQ.1) CALL ESPACE(NDLT,1,TBL(7),LRES)
      IF(LDLE.EQ.1) CALL ESPACE(NDLE,1,TBL(8),LDLE)
C------- FIND BLOCK LENGTH                                             BLLD  44
      I3=2
      I2=1+NSYM
      IF(NLBL.EQ.0) GO TO 10
      IF(NBLM.EQ.0) NBLM=NKG/NLBL+2
      GO TO 30
10    I1=NVA-IVA-(2*NBLMAX+2)/NREEL-1
      IF(I1.GE.(NKG*I2+2)) GO TO 20
C------- CASE WHERE MATRIX IS TO BE SEGMENTED
      NLBL=I1/(DEUX*I2)
      NBLM=NKG/NLBL+2
      GO TO 30
C------- CASE WHERE MATRIX IS IN CORE
20    NLBL=NKG
      NBLM=1
      I3=1
30    WRITE(MP,2010) NLBL,NBLM
2010  FORMAT(
     1  15X,'BLOCKS LENGTH IN KG                (NLBL)=',I5/
     2  15X,'MAX. NUMBER OF BLOCKS IN KG              =',I5)
      CALL ESPACE(NBLM+1,0,TBL(9),LEB)
      CALL ESPACE(NBLM,0,TBL(10),LPB)
      IF(LKGS.EQ.1) CALL ESPACE(NLBL*I3,1,TBL(1),LKGS)
      IF(NSYM.EQ.1.AND.LKGI.EQ.1) CALL ESPACE(NLBL*I3,1,TBL(3),LKGI)
      CALL EXLIND(VA(LLD),VA(LDIMP),VA(LLOCE),VA(LCORE),VA(LPRNE),
     1            VA(LPREE),VA(LNE),VA(LKE),VA(LFE),VA(LKGS),VA(LKGD),
     2            VA(LKGI),VA(LFG),VA(LCORG),VA(LDLNC),VA(LNEQ),
     3            VA(LRES),VA(LDLE),VA(LEB),VA(LPB))
      RETURN
      END
```

```
      SUBROUTINE EXLIND(KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,
     1         VKGS,VKGD,VKGI,VFG,VCORG,KDLNC,KNEQ,VRES,VDLE,KEB,KPB)
C======================================================================
C     TO EXECUTE BLOCK 'LIND'
C     ASSEMBLE AND SOLVE A LINEAR PROBLEM WHEN MATRIX KG IS STORED
C     BLOCKWISE ON DISK
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ASSE/NSYM,NKG,NKE,NDLE
      COMMON/RESO/NEQ,NRES,MRES
      COMMON/LIND/NLBL,NBLM,MKG1,MKG2
      COMMON/ES/M,MR,MP,M1,M2,M3,MDUMMY(7)
      DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),VPRNE(*),VPREE(*),
     1  KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),VCORG(*),
     2  KDLNC(*),KNEQ(*),VRES(*),VDLE(*),KEB(*),KPB(*)
C----------------------------------------------------------------------
      REWIND M3
C------- FORM TABLES EB AND PB DEFINING EQUATION BLOCKS
      CALL EQBLOC(KLD,NLBL,NBLM,NEQ,KEB,KPB)
      WRITE(MP,2000) NBLM
2000  FORMAT(15X,'NUMBER OF BLOCKS IN KG (NBLM)=',I5)
      IF(M.LT.2) GO TO 10
      I1=NBLM+1
      WRITE(MP,2010) (KEB(I),I=1,I1)
2010  FORMAT(/' FIRST EQUATION IN EACH BLOCK (EB)'/(5X,20I5))
      WRITE(MP,2020) (KPB(I),I=1,NBLM)
2020  FORMAT(/' FIRST BLOCK CONNECTED TO EACH BLOCK: (PB)'/(5X,20I5))
C------ SAVE FG UNMODIFIED FOR PRESCRIBED B.C.
10    WRITE (M3) (VFG(I),I=1,NEQ)
      IF(M.GE.2) WRITE(MP,2030) (VFG(I),I=1,NEQ)
2030  FORMAT(/' GLOBAL LOAD VECTOR UNMODIFIED FOR THE B.C. (FG)'
     1/(1X,10E12.5))
C------- ASSEMBLE KG, MODIFY FG FOR B.C. AND SAVE MODIFIED FG
      CALL ASKGD(KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,VKGS,
     1  VKGD,VKGI,VFG,VDLE,VRES,KEB)
      WRITE(M3) (VFG(I),I=1,NEQ)
C------- PRINT FG
      IF(M.GE.2) WRITE(MP,2040) (VFG(I),I=1,NEQ)
2040  FORMAT(/' GLOBAL LOAD VECTOR MODIFIED FOR THE B.C. (FG)'
     1 /(1X,10E12.5))
C
C------- SOLVE
C
20    CALL SOLD(VKGS,VKGD,VKGI,VFG,KLD,NEQ,MP,1,1,NSYM,ENERG,KEB,KPB)
      IF(NSYM.NE.1) WRITE(MP,2050) ENERG
2050  FORMAT(15X,'ENERGY    (ENERG)=',1E12.5)
C------- KG PIVOTS AND DETERMINANT
30    CALL PRPVTS(VKGD)
C------- PRINT OUT THE SOLUTION
      WRITE(MP,2060)
```

```
2060  FORMAT(//' SOLUTION'//)
      CALL PRSOL(KDLNC,VCORG,VDIMP,KNEQ,VFG)
C
C------ EVALUATE AND PRINT GRADIENTS
C
      CALL ASGRAD(KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,VKGS,
     1 VKGD,VKGI,VFG,VDLE,VRES)
C
C------ EVALUATE AND PRINT EQUILIBRIUM RESIDUALS AND REACTIONS
C
C------ READ VECTOR FG AND CHANGE ITS SIGN
      REWIND M3
      READ(M3) (VRES(I),I=1,NEQ)
      DO 40 I=1,NEQ
40    VRES(I)=-VRES(I)
C------ ASSEMBLE RESIDUALS AND REACTIONS
      CALL ASRESD(1,1,KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,
     1 VKGS,VKGD,VKGI,VFG,VDLE,VRES,VRES(NEQ+1))
C------ OUTPUT
      WRITE(MP,2070)
2070  FORMAT(//' EQUILIBRIUM RESIDUALS AND REACTIONS'//)
      CALL PRSOL(KDLNC,VCORG,VRES(NEQ+1),KNEQ,VRES)
      RETURN
      END
```

```
      SUBROUTINE EQBLOC(KLD,NLBL,NBLMAX,NEQ,KEB,KPB)
C=================================================================
C     TO FORM TABLES KEB AND KPB DEFINING EQUATION BLOCKS
C       INPUT
C          KLD     ARRAY OF A ADDRESS OF COLUMN TOP TERMS IN KG
C          NLBL    BLOCKS LENGTH
C          NBLMAX  MAX. NUMBER OF BLOCKS ALLOWED
C          NEQ     NUMBER OF EQUATIONS
C       OUTPUT
C          KEB     ARRAY CONTAINING THE NUMBERS OF FIRST EQUATIONS IN
C                  EACH BLOCK (DIMENSION  NEQ+1)
C          KPB     ARRAY CONTAINING THE NUMBER OF FIRST BLOCKS CONNECTED
C                  TO EACH BLOCK (DIMENSION NEQ)
C          NBLMAX  NUMBER OF BLOCKS
C=================================================================
      COMMON/ES/M,MR,MP,MDUMMY(10)
      DIMENSION KLD(*),KEB(*),KPB(*)
C-----------------------------------------------------------------
C------- FIRST BLOCK
      ILBL=0
      NBL=1
      KEB(1)=1
      KPB(1)=1
      IMIN=1
C------- FOR EACH EQUATION
      DO 70 IK=1,NEQ
C------- ADDRESSES FOR COLUMN IK
      JHK=KLD(IK)
      JHK1=KLD(IK+1)
      LBK1=JHK1-JHK
      IF(LBK1.LE.NLBL) GO TO 10
      WRITE(MP,2000) IK,LBK1,NLBL
2000  FORMAT(' *** ERROR,COLUMN',I5,' GREATER(',I5,')THAN BLOCK('  ,I5,'
     1)')
      STOP
C------- CHECK FOR NEW BLOCK
10    ILBL=ILBL+LBK1
      IF(ILBL.LE.NLBL) GO TO 60
      NBL=NBL+1
      IF(NBL.LE.NBLMAX) GO TO 20
      WRITE(MP,2010) IK
2010  FORMAT(' *** ERROR, EXCESSIVE NUMBER OF BLOCKS, EQUATION',I5)
      STOP
20    KEB(NBL)=IK
      ILBL=LBK1
C------- SEARCH FOR FIRST BLOCK CONNECTED TO COMPLETED BLOCK
      IB=NBL
40    IF(IMIN.GE.KEB(IB)) GO TO 50
      IB=IB-1
      GO TO 40
```

```
50    KPB(NBL-1)=IB
      IMIN=IK
C------- SEARCH FOR MINIMUM ROW NUMBER FOR COLUMN TOP TERMS
60    I=IK-LBK1+1
      IF(I.LT.IMIN)IMIN=I
70    CONTINUE
C------- FIRST BLOCK CONNECTED TO LAST BLOCK
      IB=NBL
80    IF(IMIN.GE.KEB(IB)) GO TO 90
      IB=IB-1
      GO TO 80
90    KPB(NBL)=IB
      KEB(NBL+1)=NEQ+1
      NBLMAX=NBL
      RETURN
      END
```

```
      SUBROUTINE ASKGD(KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,
     1  VKGS,VKGD,VKGI,VFG,VDLE,VRES,KEB)
C=======================================================================
C     TO ASSEMBLE GLOBAL MATRIX KG (ELEMENT FUNCTION TYPE 3)
C     TAKING INTO ACCOUNT OF PRESCRIBED NON ZERO D.O.F.
C     VERSION : MATRIX KG STORED BLOCKWISE ON FILE M4
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COND/NCLT,NCLZ,NCLNZ
      COMMON/ELEM/NELT,NNEL,NTPE,NGRE,ME,NIDENT,MNULL
      COMMON/ASSE/NSYM,MFILLR(3)
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,IPG
     1  ,ICOD,NULL(3)
      COMMON/LIND/NLBL,NBLM,MKG1,MKG2
      COMMON/ES/M,MR,MP,M1,M2,M3,M4,M5,MDUMMY(5)
      DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),VPRNE(*),VPREE(*),
     1  KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),VDLE(*),
     1  VRES(*),KEB(*)
      DATA ZERO/0.D0/
C-----------------------------------------------------------------------
C------- REWIND FILE M4
      REWIND M4
C------- LOOP OVER THE BLOCKS
      DO 80 IB=1,NBLM
C------- INITIALIZE THE BLOCK
      DO 10 I=1,NLBL
      IF(NSYM.EQ.1) VKGI(I)=ZERO
10    VKGS(I)=ZERO
      IE1=KEB(IB)
      IE2=KEB(IB+1)-1
C------- REWIND ELEMENT FILE (M2)
      REWIND M2
C------- LOOP OVER THE ELEMENTS
      DO 70 IE=1,NELT
C------- READ AN ELEMENT
      CALL RDELEM(M2,KLOCE,VCORE,VPRNE,VPREE,KNE)
C------- CHECK IF BLOCK IS AFFECTED BY THIS ELEMENT
      DO 20 ID=1,IDLE
      J=KLOCE(ID)
      IF(J.LT.IE1.OR.J.GT.IE2) GO TO 20
      GO TO 40
20    CONTINUE
30    IF(IB.NE.1.OR.(NCLNZ.EQ.0.AND.IB.EQ.1)) GO TO 70
C------- EVALUATE INTERPOLATION FUNCTIONS IF REQUIRED
40    IF(ITPE.EQ.ITPE1) GO TO 50
      ICOD=2
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C------- FORM ELEMENT MATRIX
50    ICOD=3
```

```
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C------- PRINT ELEMENT MATRIX
      IF(M.LT.2) GO TO 60
      IF(NSYM.EQ.0) IKE=IDLE*(IDLE+1)/2
      IF(NSYM.EQ.1) IKE=IDLE*IDLE
      WRITE(MP,2000) IEL,(VKE(I),I=1,IKE)
2000  FORMAT(/' MATRIX (KE) , ELEMENT:',I5/(10X,10E12.5))
C------- MODIFY FG FOR THE PRESCRIBED NON ZERO D.O.F.
60    IF(NCLNZ.NE.0.AND.IB.EQ.1) CALL MODFG(IDLE,NSYM,KLOCE,VDIMP,VKE,
     1  VFG)
C------- ASSEMBLE                                                    ASKD  61
      CALL ASSELD(1,0,IDLE,NSYM,IE1,IE2,KLOCE,KLD,VKE,VFE,VKGS,VKGD,
     1  VKGI,VFG)
      ITPE1=ITPE
70    CONTINUE
C------- END OF A BLOCK
      WRITE(M4) (VKGS(I),I=1,NLBL)
      IF(NSYM.EQ.1) WRITE(M4) (VKGI(I),I=1,NLBL)
      IF(M.LT.2) GO TO 80
      WRITE(MP,2010) IB,(VKGS(I),I=1,NLBL)
2010  FORMAT(' UPPER TRIANGLE BLOCK OF (KG) NO:',I5/(1X,10E12.5))
      IF(NSYM.EQ.1) WRITE(MP,2020) IB,(VKGI(I),I=1,NLBL)
2020  FORMAT(' LOWER TRIANGLE BLOCK OF (KG) NO:',I5/(1X,10E12.5))
80    CONTINUE
      IF(M.GE.2) WRITE(MP,2030) (VKGD(I),I=1,NEQ)
2030  FORMAT(' DIAGONAL OF (KG)'/(1X,10E12.5))
      RETURN
      END
```

```
      SUBROUTINE ASSELD(IKG,IFG,IDLE,NSYM,IE1,IE2,KLOCE,KLD,VKE,VFE,
     1  VKGS,VKGD,VKGI,VFG)
C======================================================================
C     TO ASSEMBLE ELEMENT MATRIX (SYMMETRIC OR NOT) AND/OR VECTOR.
C     THE MATRIX IS STORED BLOCKWISE ON DISK
C       INPUT
C          IKG      IF IKG.EQ.1 ASSEMBLE ELEMENT MATRIX KE
C          IFG      IF IFG.EQ.1 ASSEMBLE ELEMENT VECTOR FE
C          IDLE     NUMBER OF D.O.F. OF THE ELEMENT
C          NSYM     0=SYMMETRIC PROBLEM, 1=NON SYMMETRIC PROBLEM
C          IE1,IE2  FIRST AND LAST COLUMN OF KG TO BE ASSEMBLED
C     -    KLOCE    ELEMENT LOCALIZATION VECTOR
C          KLD      CUMULATIVE COLUMN HEIGHTS IN KG
C          VKE      ELEMENT MATRIX KE (FULL OR UPPER TRIANGLE BY
C                   DESCENDING COLUMNS)
C          VFE      ELEMENT VECTOR FE
C       OUTPUT
C          VKGS,VKGD,VKGI    GLOBAL MATRIX (SKYLINE)
C                   (SYMMETRIC OR NOT)
C          VFG      GLOBAL LOAD VECTOR
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION KLOCE(*),KLD(*),VKE(*),VFE(*),VKGS(*),VKGD(*),
     1  VKGI(*),VFG(*)
C----------------------------------------------------------------------
C
C------- ASSEMBLE ELEMENT MATRIX
C
      IF(IKG.NE.1) GO TO 100
      IOBLOC=KLD(IE1)-1
      IEQ0=IDLE
      IEQ1=1
C------- FOR EACH COLUMN OF KE
      DO 90 JD=1,IDLE
      IF(NSYM.NE.1) IEQ0=JD
      JL=KLOCE(JD)
      IF(JL) 90,90,10
10    IO=KLD(JL+1)-IOBLOC
      IEQ=IEQ1
      IQ=1
      IF(JL.LT.IE1.OR.JL.GT.IE2) GO TO 90
C------- FOR EACH ROW OF KE
      DO 80 ID=1,IDLE
      IL=KLOCE(ID)
      IF(NSYM.EQ.1) GO TO 30
      IF(ID-JD) 30,20,20
20    IQ=ID
30    IF(IL) 80,80,40
40    IJ=JL-IL
      IF(IJ) 80,50,60
```

```
C------- DIAGONAL TERMS IN KG
50    VKGD(IL)=VKGD(IL)+VKE(IEQ)
      GO TO 80
C------- UPPER TRIANGLE TERMS IN KG
60    I=IO-IJ
      VKGS(I)=VKGS(I)+VKE(IEQ)
      IF(NSYM.NE.1) GO TO 80
C------- LOWER TRIANGLE TERMS IN KG
      IEQI=(ID-1)*IDLE+JD
      VKGI(I)=VKGI(I)+VKE(IEQI)
80    IEQ=IEQ+IQ
90    IEQ1=IEQ1+IEQ0
C
C------- ASSEMBLE ELEMENT VECTOR
C
100   IF(IFG.NE.1) GO TO 130
      DO 120 ID=1,IDLE
      IL=KLOCE(ID)
      IF(IL) 120,120,110
110   VFG(IL)=VFG(IL)+VFE(ID)
120   CONTINUE
130   RETURN
      END
```

```
      SUBROUTINE SOLD(VKGS,VKGD,VKGI,VFG,KLD,NEQ,MP,IFAC,ISOL,NSYM,ENERG
     1  ,KEB,KPB)
C=======================================================================
C     TO SOLVE A LINEAR SYSTEM (SYMMETRICAL OR NOT).
C     THE MATRIX IS STORED ON FILE M4 BY SKYLINES.
C     AFTER TRIANGULARIZATION IT IS STORED ON FILE M5
C       INTPUT
C          VKGS,VKGD,VKGI    SYSTEM MATRIX : UPPER, DIAGONAL AND LOWER
C                            PARTS
C          VFG               SECOND MEMBER
C          KLD               ADDRESSES OF COLUMN TOP TERMS
C          NEQ               NUMBER OF EQUATIONS
C          MP                OUTPUT DEVICE NUMBER
C          IFAC              IF IFAC.EQ.1 TRIANGULARIZATION OF
C                            THE MATRIX
C          ISOL              IF ISOL.EQ.1 COMPUTE SOLUTION FROM THE
C                            TRIANGULARIZED MATRIX
C          NSYM              INDEX FOR NON SYMMETRIC PROBLEM
C          KEB               NUMBER OF FIRST EQUATION IN EACH
C                            BLOCK
C          KPB               NUMBER OF FIRST BLOCK CONNECTED TO EACH
C                            BLOCK
C       OUTPUT
C          VKGS,VKGD,VKGI    TRIANGULARIZED MATRIX (IF IFAC.EQ.1)
C          VFG               SOLUTION (IF ISOL.EQ.1)
C          ENERG             SYSTEM ENERGY (IF NSYM.EQ.0)
C=======================================================================
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/LIND/NLBL,NBLM,NDUMMY(2)
      COMMON/ES/M,MR,MP1,M1,M2,M3,M4,M5,MDUMMY(5)
      DIMENSION VKGS(*),VKGD(*),VKGI(*),VFG(*),KLD(*),KEB(*),KPB(*)
      DATA ZERO/0.0D0/
C-----------------------------------------------------------------------
      REWIND M4
      REWIND M5
      IK=1
      IF(VKGD(1).EQ.ZERO) GO TO 80
      ENERG=ZERO
C
C-------  FOR EACH BLOCK TO BE TRIANGULARIZED
C
      J1MIN=NLBL+1
      J1MAX=NLBL+NLBL
      DO 105 IB=1,NBLM
C-------  READ A BLOCK TO BE TRIANGULARIZED
      READ(M4) (VKGS(I),I=1,NLBL)
      IF(NSYM.EQ.1) READ(M4) (VKGI(I),I=1,NLBL)
C-------  PARAMATERS FOR BLOCK IB
      IK0=KEB(IB)
      IK1=KEB(IB+1)-1
```

```
      IBO=KPB(IB)
      JO=KLD(IKO)-1
      IF(IBO.EQ.IB) GO TO 11
C------- BACKSPACE ON CONNECTED BLOCKS
      I1=IB-IBO
      DO 10 I=1,I1
      BACKSPACE M5
      IF(NSYM.EQ.1) BACKSPACE M5
10    CONTINUE
C------- FOR EACH CONNECTED BLOCK (INCLUDING BLOCK IB ITSELF)
11    DO 103 IBC=IBO,IB
      IF(IBC.EQ.IB) GO TO 12
      READ(M5) (VKGS(I),I=J1MIN,J1MAX)
      IF(NSYM.EQ.1) READ(M5) (VKGI(I),I=J1MIN,J1MAX)
C------- PARAMETERS OF CONNECTED BLOCK
12    II0=KEB(IBC)
      II1=KEB(IBC+1)-1
      JCO=KLD(II0)-1
      IF(IBC.NE.IB) JCO=JCO-NLBL
C
C------- FOR EACH COLUMN OF BLOCK IB TO BE MODIFIED
C
      DO 100 IK=IKO,IK1
      JHK=KLD(IK)-JO
C---- ADDRESS OF NEXT COLUMN TOP TERM IK+1
      JHK1=KLD(IK+1)-JO
C---- HEIGHT OF COLUMN IK (INCLUDE UPPER AND DIAGONAL TERMS)
      LHK=JHK1-JHK
      LHK1=LHK-1
C---- ROW OF FIRST TERM TO BE MODIFIED IN COLUMN IK
      IMIN=IK-LHK1
      IMIN1=IMIN-1
C------- ROW OF LAST TERM TO BE MODIFIED IN COLUMN IK
      IMAX=IK-1
      IF(LHK1.LT.0) GO TO 100
      IF(IFAC.NE.1) GO TO 90
      IF(NSYM.EQ.0) GO TO 14
      IB1=IB
      IF(IMIN1.LT.IKO) IB1=IBO
      IF(IBC.EQ.IB1) VKGI(JHK)=VKGI(JHK)/VKGD(IMIN1)
14    IF(IBC.EQ.IB.AND.IK.EQ.IKO) GO TO 40
      IF(LHK1.EQ.0) GO TO 40
C------- FIND FIRST AND LAST ROW OF COLUMN IK AFFECTED
C         BY CONNECTED BLOCK IBC
      IMINC=MAXO(IMIN,II0)
      IMAXC=MINO(IMAX,II1)
      IF(IMINC.GT.IMAXC) GO TO 40
C
C---- MODIFY NON DIAGONAL TERMS OF COLUMN IK
C
      JCK=JHK+IMINC-IMIN1
```

```
      JHJ=KLD(IMINC)-JCO
C------   FOR EACH TERM TO BE MODIFIED, LOCATED AT JCK
      DO 30 IJ=IMINC,IMAXC
      JHJ1=KLD(IJ+1)-JCO
C-----   NUMBER OF MODIFICATIVE TERMS OF COEFFICIENT LOCATED AT JCK
      IC=MINO(JCK-JHK,JHJ1-JHJ)
      IF(IC.LE.0.AND.NSYM.EQ.0) GO TO 20
      C1=ZERO
      IF(IC.LE.0) GO TO 17
      J1=JHJ1-IC
      J2=JCK-IC
      IF(NSYM.EQ.1) GO TO 15
      VKGS(JCK)=VKGS(JCK)-SCAL(VKGS(J1),VKGS(J2),IC)
      GO TO 20
15    VKGS(JCK)=VKGS(JCK)-SCAL(VKGI(J1),VKGS(J2),IC)
      C1=SCAL(VKGS(J1),VKGI(J2),IC)
17    VKGI(JCK)=(VKGI(JCK)-C1)/VKGD(IJ)
20    JCK=JCK+1
30    JHJ=JHJ1
C
C-----   MODIFY DIAGONAL TERM
C
40    IF(IBC.NE.IB) GO TO 90
      JCK=JHK
      CDIAG=ZERO
      DO 70 IJ=IMIN1,IMAX
      C1=VKGS(JCK)
      IF(NSYM.EQ.1) GO TO 50
      C2=C1/VKGD(IJ)
      VKGS(JCK)=C2
      GO TO 60
50    C2=VKGI(JCK)
60    CDIAG=CDIAG+C1*C2
70    JCK=JCK+1
      VKGD(IK)=VKGD(IK)-CDIAG
      IF(VKGD(IK)) 90,80,90
80    WRITE(MP,2000) IK
2000  FORMAT(' *** ERROR, ZERO PIVOT EQUATION ',I5)
      STOP
C
C-----   SOLVE LOWER TRIANGULAR SYSTEM
C
   90 IF(ISOL.NE.1) GO TO 100
      IF(IBC.NE.IB) GO TO 100
      IF(NSYM.NE.1) VFG(IK)=VFG(IK)-SCAL(VKGS(JHK),VFG(IMIN1),LHK)
      IF(NSYM.EQ.1) VFG(IK)=VFG(IK)-SCAL(VKGI(JHK),VFG(IMIN1),LHK)
100   CONTINUE
C-------   NEXT CONNECTED BLOCK
103   CONTINUE
C-------   END OF ELIMINATION OF THIS BLOCK
      IF(IB.EQ.NBLM) GO TO 105
```

```
      WRITE(M5) (VKGS(I),I=1,NLBL)
      IF(NSYM.EQ.1)  WRITE(M5) (VKGI(I),I=1,NLBL)
 105  CONTINUE
      IF(ISOL.NE.1) RETURN
C
C----- SOLVE DIAGONAL SYSTEM
C
      IF(NSYM.EQ.1) GO TO 120
      DO 110 IK=1,NEQ
      C1=VKGD(IK)
      C2=VFG(IK)/C1
      VFG(IK)=C2
 110  ENERG=ENERG+C1*C2*C2
C
C----- SOLVE UPPER TRIANGULAR SYSTEM
C
 120  IB=NBLM
      IK0=KEB(IB)-1
      J0=KLD(IK0+1)-1
      IK=NEQ+1
      JHK1=KLD(IK)-J0
C------- FOR EVERY EQUATION FROM NEQ TO 1
 130  IK=IK-1
C------- READ A BLOCK IF REQUIRED
      IF(IK.NE.IK0) GO TO 135
      BACKSPACE M5
      IF(NSYM.EQ.1) BACKSPACE M5
      READ(M5) (VKGS(I),I=1,NLBL)
      IF(NSYM.EQ.1) READ(M5) (VKGI(I),I=1,NLBL)
      BACKSPACE M5
      IF(NSYM.EQ.1) BACKSPACE M5
      IB=IB-1
      IK0=KEB(IB)-1
      J0=KLD(IK0+1)-1
      JHK1=KLD(IK+1)-J0
C------- MODIFY THE UNKNOWN VECTOR
 135  IF(NSYM.EQ.1) VFG(IK)=VFG(IK)/VKGD(IK)
      IF(IK.EQ.1) RETURN
      C1=VFG(IK)
      JHK=KLD(IK)-J0
      JBK=JHK1-1
      IF(JHK.GT.JBK)GO TO 150
      IJ=IK-JBK+JHK-1
      DO 140 JCK=JHK,JBK
      VFG(IJ)=VFG(IJ)-VKGS(JCK)*C1
 140  IJ=IJ+1
 150  JHK1=JHK
      GO TO 130
      END
```

```
      SUBROUTINE BLNLIN
C=====================================================================
C     TO CALL BLOCK 'NLIN'
C     TO SOLVE A STEADY NON LINEAR PROBLEM
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/ELEM/NULL(4),ME,MNULL(2)
      COMMON/ASSE/NSYM,NKG,NKE,NDLE
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/NLIN/EPSDL,XNORM,OMEGA,XPAS,DPAS,DPAS0,NPAS,IPAS,NITER,
     1  ITER,IMETH
      COMMON/ES/M,MR,MP,M1,M2,M3,M4,MDUMMY(6)
      COMMON/LOC/LCORG,LDLNC,LNEQ,LDIMP,LPRNG,LPREG,LLD,LLOCE,LCORE,LNE,
     1  LPRNE,LPREE,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,LRES,LDLG,LME,
     2  LDUMMY(3)
      COMMON VA(1)
      DIMENSION TBL(10),IN(2),XIN(3)
C+++     THIS IS COMMENTED OUT BECAUSE OF AN MS FORTRAN COMPILER
C+++     BUG WHICH WILL NOT INITIALIZE  $LARGE ARRAYS.  THIS ARRAY
C+++     IS NOW INITIALIZED BY A CALL TO A DUMMY SUBROUTINE
C+++     INITBL WHICH EXISTS SOLELY TO INITIALIZE THIS ARRAY
C
C     DATA TBL/'KGS ','KGD ','KGI ','FG  ','KE  ','FE  ','RES ','DLE ',
C    * 'DLG ','ME  '/
C
C        HERE IS THE CALL TO GET AROUND THE COMPILER BUG
      CALL INITBL(TBL,'NLIN')
C
C+++     ALL OF THIS IS TO GET AROUND THE MICROSOFT
C+++     COMPILER BUG
C
C---------------------------------------------------------------------
      IF(M1.EQ.0) M1=MR
      IF(M2.EQ.0) M2=ME
      WRITE(MP,2000) M
2000  FORMAT(//' NON LINEAR SOLUTION (M=',I2,')'/1X,23('='))
C------- TO ALLOCATE SPACE
      IF(LKGS.EQ.1) CALL ESPACE(NKG,1,TBL(1),LKGS)
      IF(LKGD.EQ.1) CALL ESPACE(NEQ,1,TBL(2),LKGD)
      IF(NSYM.EQ.1.AND.LKGI.EQ.1) CALL ESPACE(NKG,1,TBL(3),LKGI)
      IF(LFG.EQ.1) CALL ESPACE (NEQ,1,TBL(4),LFG)
      IF(LKE.EQ.1) CALL ESPACE(NKE,1,TBL(5),LKE)
      IF(LFE.EQ.1) CALL ESPACE(NDLE,1,TBL(6),LFE)
      IF(LRES.EQ.1) CALL ESPACE(NEQ,1,TBL(7),LRES)
      IF(LDLE.EQ.1) CALL ESPACE(NDLE,1,TBL(8),LDLE)
      IF(LDLG.EQ.1) CALL ESPACE(NEQ,1,TBL(9),LDLG)
      IF(LME.EQ.1) CALL ESPACE(NKE,1,TBL(10),LME)
C------- TO EXECUTE THE BLOCK
      CALL EXNLIN(VA(LCORG),VA(LDLNC),VA(LDIMP),VA(LNEQ),VA(LLD),
```

```
1  VA(LLOCE),VA(LCORE),VA(LPRNE),VA(LPREE),VA(LNE),VA(LKE),VA(LME),
2  VA(LFE),VA(LDLE),VA(LKGS),VA(LKGD),VA(LKGI),VA(LFG),VA(LREE),
3  VA(LDLG))
   RETURN
   END
```

```
$LARGE
$NOFLOATCALLS
      SUBROUTINE EXNLIN(VCORG,KDLNC,VDIMP,KNEQ,KLD,KLOCE,VCORE,VPRNE,
     1  VPREE,KNE,VKE,VME,VFE,VDLE,VKGS,VKGD,VKGI,VFG,VRES,VDLG)
C=======================================================================
C     TO EXECUTE BLOCK 'NLIN'
C     TO SOLVE A STEADY NON LINEAR PROBLEM
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/COND/NCLT,NCLZ,NCLNZ
      COMMON/ASSE/NSYM,MFILLR(3)
      COMMON/NLIN/EPSDL,XNORM,OMEGA,XPAS,DPAS,DPAS0,NPAS,IPAS,NITER,
     1  ITER,IMETH
      COMMON/ES/M,MR,MP,M1,M2,M3,M4,MDUMMY(6)
      DIMENSION VCORG(*),KDLNC(*),VDIMP(*),KNEQ(*),KLD(*),KLOCE(*),
     1  VCORE(*),VPRNE(*),VPREE(*),KNE(*),VKE(*),VME(*),VFE(*),VDLE(*),
     2  VKGS(*),VKGD(*),VKGI(*),VFG(*),VRES(*),VDLG(*)
      DATA ZERO/0.D0/
C-----------------------------------------------------------------------
      DPAS0=ZERO
      XPAS=ZERO
      IPAS=0
C------- READY INITIAL D.O.F. ON FILE M3
      IF(M3.EQ.0) GO TO 10
      REWIND M3
      READ(M3) (VDLG(I),I=1,NEQ)
C------- READ A CARD DEFINING A SET OF IDENTICAL STEPS
10    READ(M1,1000) DPAS,I1,I2,I3,X1,X2
1000  FORMAT(F10.0,3I5,2F10.0)
      IF(DPAS.EQ.ZERO) GO TO 140
      IF(I1.GT.0) NPAS=I1
      IF(I2.GT.0) NITER=I2
      IF(I3.GT.0) IMETH=I3
      IF(X1.GT.ZERO) EPSDL=X1
      IF(X2.GT.ZERO) OMEGA=X2
C
C------- LOOP OVER ALL STEPS
C
      DO 130 IP=1,NPAS
      IPAS=IPAS+1
      XPAS=XPAS+DPAS
      WRITE(MP,2000) IPAS,DPAS,XPAS,NITER,IMETH,EPSDL,OMEGA
2000  FORMAT(/1X,13('-'),'STEP NUMBER (IPAS):',I5//
     1            14X,'INCREMENT                (DPAS)=',E12.5/
```

```
     2              14X,'TOTAL LEVEL                (XPAS)=',E12.5/
     3              14X,'NUMBER OF ITERATIONS       (NITER)=',I12/
     4              14X,'METHOD NUMBER              (IMETH):',I12/
     5              14X,'TOLERANCE                  (EPSDL)=',E12.5/
     6              14X,'OVER RELAXATION FACTOR     (OMEGA)=',E12.5/)
C
C------- LOOP OVER EQUILIBRIUM ITERATIONS
C
      DO 110 ITER=1,NITER
C------- CHOOSE THE METHOD
      IF(IMETH.GT.3) GO TO 20
C------- NEWTON TYPE METHODS
      CALL NEWTON(VCORG,KDLNC,VDIMP,KNEQ,KLD,KLOCE,VCORE,VPRNE,VPREE,
     1  KNE,VKE,VME,VFE,VDLE,VKGS,VKGD,VKGI,VFG,VRES,VDLG)
      GO TO 100
C------- OTHER METHODS ......
20    CONTINUE
      WRITE(MP,2010) IMETH
2010  FORMAT(' ** ERROR, METHOD:',I3,' UNKNOWN')
      STOP
C------- COMPUTE THE NORM
100   CALL NORME(NEQ,VRES,VDLG,XNORM)
      IF(M.GT.0) WRITE(MP,2020) ITER,XNORM
2020  FORMAT(5X,'ITERATION (ITER):',I3,' NORM (XNORM)=',E12.5)
      IF(M.GE.2) CALL PRSOL(KDLNC,VCORG,VDIMP,KNEQ,VDLG)
      IF(XNORM.LE.EPSDL) GO TO 120
110   CONTINUE
      ITER=NITER
C------- END OF STEP
120   DPASO=DPAS
      WRITE(MP,2030) ITER,NITER
2030  FORMAT(/10X,I4,' PERFORMED ITERATIONS OVER',I4/)
      IF(M.LT.2) CALL PRSOL(KDLNC,VCORG,VDIMP,KNEQ,VDLG)
130   CONTINUE
      GO TO 10
C------- SAVE THE SOLUTION ON FILE M4
140   IF(M4.NE.0) WRITE(M4) (VDLG(I),I=1,NEQ)
      RETURN
      END
```

```
      SUBROUTINE NEWTON(VCORG,KDLNC,VDIMP,KNEQ,KLD,KLOCE,VCORE,VPRNE,
     1  VPREE,KNE,VKE,VME,VFE,VDLE,VKGS,VKGD,VKGI,VFG,VRES,VDLG)
C=======================================================================
C     ALGORITHM FOR NEWTON-RAPHSON TYPE METHODS
C       IMETH.EQ.1  COMPUTE K AT EACH ITERATION
C       IMETH.EQ.2  K IS CONSTANT
C       IMETH.EQ.3  RECOMPUTE K AT THE BEGINNING OF EACH STEP
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ASSE/NSYM,NKG,MFILLR(2)
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/NLIN/EPSDL,XNORM,OMEGA,XPAS,DPAS,DPASO,NPAS,IPAS,NITER,
     1  ITER,IMETH
      COMMON/ES/M,MR,MP,MDUMMY(10)
      DIMENSION VCORG(*),KDLNC(*),VDIMP(*),KNEQ(*),KLD(*),KLOCE(*),
     1  VCORE(*),VPRNE(*),VPREE(*),KNE(*),VKE(*),VME(*),VFE(*),VDLE(*),
     2  VKGS(*),VKGD(*),VKGI(*),VFG(*),VRES(*),VDLG(*)
      DATA ZERO/0.D0/,UN/1.D0/
C-----------------------------------------------------------------------
C------- DECIDE IF GLOBAL MATRIX IS TO REASSEMBLED
      IKT=0
      IF(IMETH.EQ.1) GO TO 10
      IF(IPAS.EQ.1.AND.ITER.EQ.1) GO TO 10
      IF(IMETH.EQ.3.AND.ITER.EQ.1) GO TO 10
      GO TO 20
10    IKT=1
C------- INITIALIZE GLOBAL MATRIX TO ZERO IF IT IS TO BE ASSEMBLED
20    IF(IKT.EQ.0)GO TO 30
      CALL INIT(ZERO,NKG,VKGS)
      CALL INIT(ZERO,NEQ,VKGD)
      IF(NSYM.EQ.1) CALL INIT(ZERO,NKG,VKGI)
C------- STORE LOADS IN THE RESIDUAL VECTOR
30    CALL MAJ(XPAS,ZERO,NEQ,VFG,VRES)
C------- ASSEMBLE RESIDUAL VECTOR, AND EVENTUALLY THE GLOBAL MATRIX
      CALL ASNEWT(IKT,KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,
     1  VKGS,VKGD,VKGI,VDLG,VDLE,VRES)
C------- SOLVE
      CALL SOL(VKGS,VKGD,VKGI,VRES,KLD,NEQ,MP,IKT,1,NSYM,ENERG)
      IF(IKT.EQ.1.AND.M.GT.1) CALL PRPVTS(VKGD)
C------- UPDATE THE SOLUTION                                        NEWT  40
      CALL MAJ(OMEGA,UN,NEQ,VRES,VDLG)
      RETURN
      END
```

```
      SUBROUTINE ASNEWT(IKT,KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,
     1  KNE,VKE,VFE,VKGS,VKGD,VKGI,VFG,VDLE,VRES)
C=================================================================
C     TO ASSEMBLE THE RESIDUALS AND THE GLOBAL MATRIX (IF IKT.EQ.1)
C     WHILE LOOPING OVER THE ELEMENTS
C     (FOR THE NEWTON-RAPHSON METHOD)
C=================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ELEM/NELT,NNEL,NTPE,NGRE,ME,NIDENT,MNULL
      COMMON/ASSE/NSYM,MFILLR(3)
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,IPG
     1  ,ICOD,NULL(3)
      COMMON/ES/M,MR,MP,M1,M2,MDUMMY(8)
      DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),VPRNE(*),VPREE(*),
     1  KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),VDLE(*),
     2  VRES(*)
C-----------------------------------------------------------------
C------- REWIND ELEMENT FILE M2
      REWIND M2
C------- LOOP OVER THE ELEMENTS
      DO 40 IE=1,NELT
C------- READ AN ELEMENT
      CALL RDELEM(M2,KLOCE,VCORE,VPRNE,VPREE,KNE)
C------- EVALUATE INTERPOLATION FUNCTIONS IF REQUIRED
      IF(ITPE.EQ.ITPE1) GO TO 10
      ICOD=2
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C------- FIND THE D.O.F. OF THE ELEMENT FROM VFG
10    CALL DLELM(KLOCE,VFG,VDIMP,VDLE)
C------- CALCULATE ELEMENT RESIDUALS AND CHANGE THEIR SIGN
      ICOD=6
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
      DO 20 I=1,IDLE
20    VFE(I)=-VFE(I)
C------- EVALUATE GLOBAL MATRIX
      IF(IKT.EQ.0) GO TO 30
      ICOD=4
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C------- ASSEMBLE THE RESIDUALS AND THE GLOBAL MATRIX
30    CALL ASSEL(IKT,1,IDLE,NSYM,KLOCE,KLD,VKE,VFE,VKGS,VKGD,VKGI,VRES)
40    ITPE1=ITPE
      RETURN
      END
```

```
      SUBROUTINE INIT(X,N,V)
C=======================================================================
C     INITIALIZE VECTOR V TO VALUE X
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION V(*)
C-----------------------------------------------------------------------
      DO 10 I=1,N
10    V(I)=X
      RETURN
      END


      SUBROUTINE MAJ(X1,X2,N,V1,V2)
C=======================================================================
C     EXECUTE THE VECTOR OPERATION: V2=X1*V1 + X2*V2
C       X1,X2:SCALARS   V1,V2:VECTORS
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION V1(*),V2(*)
C-----------------------------------------------------------------------
      DO 10 I=1,N
10    V2(I)=X1*V1(I)+X2*V2(I)
      RETURN
      END


      SUBROUTINE NORME(N,VDEL,V,XNORM)
C=======================================================================
C     COMPUTE THE LENGTHS RATIO OF VECTORS VDEL AND V
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION VDEL(*),V(*)
      DATA ZERO/0.D0/,UN/1.D0/,FAC/1.D-3/
      SQRT(X)=DSQRT(X)
C-----------------------------------------------------------------------
      C1=ZERO
      C2=ZERO
      DO 10 I=1,N
      C1=C1+VDEL(I)*VDEL(I)
10    C2=C2+V(I)*V(I)
      C=C1*FAC
      IF(C2.LE.C) C2=UN
      XNORM=SQRT(C1/C2)
      RETURN
      END
```

```
      SUBROUTINE BLTEMP
C=====================================================================
C     TO CALL BLOCK 'TEMP'
C     TO SOLVE AN UNSTEADY PROBLEM (LINEAR OR NOT)
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/ELEM/NULL(4),ME,MNULL(2)
      COMMON/ASSE/NSYM,NKG,NKE,NDLE
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/NLIN/EPSDL,XNORM,OMEGA,XPAS,DPAS,DPASO,NPAS,IPAS,NITER,
     1 ITER,IMETH
      COMMON/ES/M,MR,MP,M1,M2,M3,M4,MDUMMY(5)
      COMMON/LOC/LCORG,LDLNC,LNEQ,LDIMP,LPRNG,LPREG,LLD,LLOCE,LCORE,LNE,
     1 LPRNE,LPREE,LDLE,LKE,LFE,LKGS,LKGD,LKGI,LFG,LRES,LDLG,LME,
     1 LDLEO,LDLGO,LFGO
      COMMON VA(1)
      DIMENSION TBL(13),IN(2),XIN(3)
C+++     THIS IS COMMENTED OUT BECAUSE OF AN MS FORTRAN COMPILER
C+++     BUG WHICH WILL NOT INITIALIZE  $LARGE ARRAYS.  THIS ARRAY
C+++     IS NOW INITIALIZED BY A CALL TO A DUMMY SUBROUTINE
C+++     INITBL WHICH EXISTS SOLELY TO INITIALIZE THIS ARRAY
C
C     DATA TBL/'KGS ','KGD ','KGI ','FG  ','KE  ','FE  ','RES ',
C    * 'DLE ','DLG ','ME  ','DLEO','DLGO','FGO '/
C
C        HERE IS THE CALL TO GET AROUND THE COMPILER BUG
      CALL INITBL(TBL,'TEMP')
C
C+++     ALL OF THIS IS TO GET AROUND THE MICROSOFT
C+++     COMPILER BUG
C
C---------------------------------------------------------------------
      IF(M1.EQ.0) M1=MR
      IF(M2.EQ.0) M2=ME
      WRITE(MP,2000) M
2000  FORMAT(//' UNSTEADY SOLUTION (M=',I2,')'/1X,23('='))
C------- TO ALLOCATE SPACE
      IF(LKGS.EQ.1) CALL ESPACE(NKG,1,TBL(1),LKGS)
      IF(LKGD.EQ.1) CALL ESPACE(NEQ,1,TBL(2),LKGD)
      IF(NSYM.EQ.1.AND.LKGI.EQ.1) CALL ESPACE(NKG,1,TBL(3),LKGI)
      IF(LFG.EQ.1) CALL ESPACE (NEQ,1.TBL(4),LFG)
      IF(LKE.EQ.1) CALL ESPACE(NKE,1,TBL(5),LKE)
      IF(LFE.EQ.1) CALL ESPACE(NDLE,1,TBL(6),LFE)
      IF(LRES.EQ.1) CALL ESPACE(NEQ,1,TBL(7),LRES)
      IF(LDLE.EQ.1) CALL ESPACE(NDLE,1,TBL(8),LDLE)
      IF(LDLG.EQ.1) CALL ESPACE(NEQ,1,TBL(9),LDLG)
      IF(LME.EQ.1) CALL ESPACE(NKE,1,TBL(10),LME)
      IF(LDLEO.EQ.1) CALL ESPACE(NDLE,1,TBL(11),LDLEO)
      IF(LDLGO.EQ.1) CALL ESPACE(NEQ,1,TBL(12),LDLGO)
```

```
      IF(LFG0.EQ.1) CALL ESPACE(NEQ,1,TBL(13),LFG0)
C------- TO EXECUTE THE BLOCK
      CALL EXTEMP(VA(LCORG),VA(LDLNC),VA(LDIMP),VA(LNEQ),VA(LLD),
     1 VA(LLOCE),VA(LCORE),VA(LPRNE),VA(LPREE),VA(LNE),VA(LKE),VA(LME),
     2 VA(LFE),VA(LDLE),VA(LKGS),VA(LKGD),VA(LKGI),VA(LFG),VA(LRES),
     3 VA(LDLG),VA(LDLE0),VA(LDLG0),VA(LFG0))
      RETURN
      END


      SUBROUTINE EXTEMP(VCORG,KDLNC,VDIMP,KNEQ,KLD,KLOCE,VCORE,VPRNE,
     1 VPREE,KNE,VKE,VME,VFE,VDLE,VKGS,VKGD,VKGI,VFG,VRES,VDLG,
     2 VDLE0,VDLG0,VFG0)
C======================================================================
C     TO EXECUTE BLOCK 'TEMP'
C     TO SOLVE AN UNSTEADY PROBLEM (LINEAR OR NOT)
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/COND/NCLT,NCLZ,NCLNZ
      COMMON/ASSE/NSYM,MFILLR(3)
      COMMON/NLIN/EPSDL,XNORM,OMEGA,XPAS,DPAS,DPAS0,NPAS,IPAS,NITER,
     1 ITER,IMETH
      COMMON/ES/M,MR,MP,M1,M2,M3,M4,MDUMMY(6)
      DIMENSION VCORG(*),KDLNC(*),VDIMP(*),KNEQ(*),KLD(*),KLOCE(*),
     1 VCORE(*),VPRNE(*),VPREE(*),KNE(*),VKE(*),VME(*),VFE(*),VDLE(*),
     2 VKGS(*),VKGD(*),VKGI(*),VFG(*),VRES(*),VDLG(*),VDLE0(*),
     3 VDLG0(*),VFG0(*)
      DATA ZERO/0.D0/,UN/1.D0/
C----------------------------------------------------------------------
      DPAS0=ZERO
      XPAS=ZERO
      IPAS=0
C------- READ INITIAL D.O.F. ON FILE M3
      IF(M3.EQ.0) GO TO 5
      REWIND M3
      READ(M3) (VDLG(I),I=1,NEQ)
      CALL MAJ(UN,ZERO,NEQ,VDLG,VDLG0)
C------- SAVE THE REFERENCE LOAD CONDITIONS
5     CALL MAJ(UN,ZERO,NEQ,VFG,VFG0)
C------- READ A CARD DEFINING A SET OF IDENTICAL STEPS
10    READ(M1,1000) DPAS,I1,I2,I3,X1,X2
1000  FORMAT(F10.0,3I5,2F10.0)
      IF(DPAS.EQ.ZERO) GO TO 140
      IF(I1.GT.0) NPAS=I1
      IF(I2.GT.0) NITER=I2
      IF(I3.GT.0) IMETH=I3
      IF(X1.GT.ZERO) EPSDL=X1
      IF(X2.NE.ZERO) OMEGA=X2
C
C------- LOOP OVER THE STEPS
```

```
C
      DO 130 IP=1,NPAS
      CALL INIT(ZERO,NEQ,VFG)
      IPAS=IPAS+1
      XPAS=XPAS+DPAS
      WRITE(MP,2000) IPAS,DPAS,XPAS,NITER,IMETH,EPSDL,OMEGA
2000  FORMAT(/1X,13('-'),'STEP NUMBER (IPAS):',I5//
     1             14X,'INCREMENT                 (DPAS)=',E12.5/
     2             14X,'TOTAL LEVEL               (XPAS)=',E12.5/
     3             14X,'NUMBER OF ITERATIONS     (NITER)=',I12/
     4             14X,'METHOD NUMBER            (IMETH):',I12/
     5             14X,'TOLERANCE                (EPSDL)=',E12.5/
     6             14X,'COEFFICIENT ALPHA        (OMEGA)=',E12.5/)
C
C------- LOOP OVER EQUILIBRIUM ITERATIONS
C
      DO 110 ITER=1,NITER
C------- CHOOSE THE METHOD
      IF(IMETH.GT.3) GO TO 20
C------- EULER TYPE METHODS
      CALL EULER(VCORG,KDLNC,VDIMP,KNEQ,KLD,KLOCE,VCORE,VPRNE,VPREE,
     1 KNE,VKE,VME,VFE,VDLE,VKGS,VKGD,VKGI,VFG,VRES,VDLG,
     2 VDLE0,VDLG0,VFG0)
      GO TO 100
C------- OTHER METHODS ......
20    CONTINUE
      WRITE(MP,2010) IMETH
2010  FORMAT(' ** ERROR, METHOD:',I3,' UNKNOWN')
      STOP
C------- COMPUTE THE NORM
100   CALL NORME(NEQ,VRES,VDLG,XNORM)
      IF(M.GT.0) WRITE(MP,2020) ITER,XNORM
2020  FORMAT(5X,'ITERATION (ITER):',I3,' NORM (XNORM)=',E12.5)
      IF(M.GE.2) CALL PRSOL(KDLNC,VCORG,VDIMP,KNEQ,VDLG)
      IF(XNORM.LE.EPSDL) GO TO 120
110   CONTINUE
C------- END OF STEP
120   DPAS0=DPAS
      CALL MAJ(UN,ZERO,NEQ,VDLG,VDLG0)
      CALL PRSOL(KDLNC,VCORG,VDIMP,KNEQ,VDLG)
130   CONTINUE
      GO TO 10
C------- SAVE THE SOLUTION ON FILE M4
140   IF(M4.NE.0) WRITE(M4) (VDLG(I),I=1,NEQ)
      RETURN
      END
```

```
      SUBROUTINE EULER(VCORG,KDLNC,VDIMP,KNEQ,KLD,KLOCE,VCORE,VPRNE,
     1 VPREE,KNE,VKE,VME,VFE,VDLE,VKGS,VKGD,VKGI,VFG,VRES,VDLG,
     2 VDLEO,VDLGO,VFGO)
C=================================================================
C     ALGORITHM FOR EULER TYPE METHODS (IMPLICIT, EXPLICIT OR BOTH
C     ACCORDING TO OMEGA) FOR LINEAR OR NON LINEAR PROBLEMS.
C     THE NON LINEAR PROBLEM IS SOLVED BY A NEWTON-RAPHSON
C     METHOD
C       IMETH.EQ.1  STANDARD NEWTON-RAPHSON
C       IMETH.EQ.2  K IS CONSTANT
C       IMETH.EQ.3  K IS RECOMPUTED AT THE BEGINNING OF EACH STEP
C=================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ASSE/NSYM,NKG,MFILLR(2)
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/NLIN/EPSDL,XNORM,OMEGA,XPAS,DPAS,DPASO,NPAS,IPAS,NITER,
     1 ITER,IMETH
      COMMON/ES/M,MR,MP,MDUMMY(10)
      DIMENSION VCORG(*),KDLNC(*),VDIMP(*),KNEQ(*),KLD(*),KLOCE(*),
     1 VCORE(*),VPRNE(*),VPREE(*),KNE(*),VKE(*),VME(*),VFE(*),
     2 VDLE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),VRES(*),VDLG(*),
     3 VDLEO(*),VDLGO(*),VFGO(*)
      DATA ZERO/0.D0/,UN/1.D0/
C-----------------------------------------------------------------
C------- DECIDE IF GLOBAL MATRIX IS TO BE REASSEMBLED
      IKT=0
      IF(IMETH.EQ.1) GO TO 10
      IF(DPAS.NE.DPASO.AND.ITER.EQ.1) GO TO 10
      IF(IMETH.EQ.3.AND.ITER.EQ.1) GO TO 10
      GO TO 20
10    IKT=1
C------- INITIALIZE GLOBAL MATRIX TO ZERO IF NECESSARY
20    IF(IKT.EQ.0) GO TO 30
      CALL INIT(ZERO,NKG,VKGS)
      CALL INIT(ZERO,NEQ,VKGD)
      IF(NSYM.EQ.1) CALL INIT(ZERO,NKG,VKGI)
C------- ASSEMBLE RESIDUALS AND GLOBAL MATRIX IF REQUIRED
30    CALL MAJ(UN,ZERO,NEQ,VFGO,VRES)
      CALL ASEULR(IKT,VCORG,KDLNC,VDIMP,KNEQ,KLD,KLOCE,VCORE,VPRNE,
     1 VPREE,KNE,VKE,VME,VFE,VDLE,VKGS,VKGD,VKGI,VFG,VRES,VDLG,
     2 VDLEO,VDLGO,VFGO)
      C1=UN
      IF(ITER.GT.1) C1=C1-OMEGA
      DO 40 I=1,NEQ
40    VRES(I)=DPAS*(VRES(I)-C1*VFG(I))
C------- SOLVE
      CALL SOL(VKGS,VKGD,VKGI,VRES,KLD,NEQ,MP,IKT,1,NSYM,ENERG)
C------- UPDATE THE SOLUTION
      CALL MAJ(UN,UN,NEQ,VRES,VDLG)
      RETURN
      END
```

```
      SUBROUTINE ASEULR(IKT,VCORG,KDLNC,VDIMP,KNEQ,KLD,KLOCE,VCORE,
     1  VPRNE,VPREE,KNE,VKE,VME,VFE,VDLE,VKGS,VKGD,VKGI,VFG,VRES,
     2  VDLG,VDLE0,VDLG0,VFG0)
C=====================================================================
C     TO ASSEMBLE THE RESIDUALS AND THE GLOBAL MATRIX (IF IKT.EQ.1)
C     WHILE LOOPING OVER THE ELEMENTS (FOR EULER METHOD)
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ELEM/NELT,NNEL,NTPE,NGRE,ME,NIDENT,MNULL
      COMMON/ASSE/NSYM,MFILLR(3)
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,IPG
     1  ,ICOD,NULL(3)
      COMMON/NLIN/EPSDL,XNORM,OMEGA,XPAS,DPAS,DPAS0,NPAS,IPAS,NITER,
     1  ITER,IMETH
      COMMON/ES/M,MR,MP,M1,M2,MDUMMY(8)
      DIMENSION VCORG(*),KDLNC(*),VDIMP(*),KNEQ(*),KLD(*),KLOCE(*),
     1  VCORE(*),VPRNE(*),VPREE(*),KNE(*),VKE(*),VME(*),VFE(*),VDLE(*),
     2  VKGS(*),VKGD(*),VKGI(*),VFG(*),VRES(*),VDLG(*),VDLE0(*),
     3  VDLG0(*),VFG0(*)
      DATA UN/1.D0/
C---------------------------------------------------------------------
      CC=DPAS*OMEGA
      IFE=0
      IF(ITER.GT.1) IFE=1
C------- REWIND ELEMENT FILE (ME)
      REWIND M2
C------- LOOP OVER THE ELEMENTS
      DO 90 IE=1,NELT
C------- READ AN ELEMENT
      CALL RDELEM(M2,KLOCE,VCORE,VPRNE,VPREE,KNE)
C------- EVALUATE INTERPOLATION FUNCTIONS IF REQUIRED
      IF(ITPE.EQ.ITPE1) GO TO 10
      ICOD=2
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C------- FIND ELEMENT D.O.F. FROM VFG
10    CALL DLELM(KLOCE,VDLG,VDIMP,VDLE)
C------- COMPUTE THE RESIDUAL K.U.
      ICOD=6
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C------- COMPUTE MATRIX M
      ICOD=5
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VME,VFE)
C------- COMPUTE MATRIX K IF REQUIRED
      IF(IKT.EQ.0) GO TO 15
      ICOD=3
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C------- RESIDUALS OF THE FIRST ITERATION IN EACH STEP (LINEAR)
15    IF(ITER.GT.1) GO TO 20
      CALL ASSEL(0,1,IDLE,NSYM,KLOCE,KLD,VKE,VFE,VKGS,VKGD,VKGI,VFG)
```

```
      GO TO 60
C------  RESIDUALS AFTER FIRST ITERATION
20    CALL DLELM(KLOCE,VDLG0,VDIMP,VDLE0)
      DO 30 I=1,IDLE
      VDLE(I)=(VDLE0(I)-VDLE(I))/DPAS
30    VFE(I)=-OMEGA*VFE(I)
C------  PRODUCT M . U
      VFE(1)=VFE(1)+VME(1)*VDLE(1)
      II=1
      DO 50 J=2,IDLE
      J1=J-1
      DO 40 I=1,J1
      II=II+1
      VFE(I)=VFE(I)+VME(II)*VDLE(J)
40    VFE(J)=VFE(J)+VME(II)*VDLE(I)
      II=II+1
50    VFE(J)=VFE(J)+VME(II)*VDLE(J)
C-------  MATRIX    M + DPAS.OMEGA. K
60    IF(IKT.EQ.0) GO TO 80
      II=0
      DO 70 I=1,IDLE
      DO 70 J=I,IDLE
      II=II+1
70    VKE(II)=VKE(II)*CC+VME(II)
C-------  ASSEMBLE THE RESIDUAL AND THE GLOBAL MATRIX
80    CALL ASSEL(IKT,IFE,IDLE,NSYM,KLOCE,KLD,VKE,VFE,VKGS,VKGD,VKGI,
     1  VRES)
90    ITPE1=ITPE
      RETURN
      END
```

```
      SUBROUTINE BLVALP
C=======================================================================
C     TO CALL BLOCK 'VALP'
C     TO COMPUTE EIGENVALUES AND EIGENVECTORS BY THE SUBSPACE
C     ITERATION TECHNIQUE
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*4 TBL
      COMMON/ELEM/NULL(4),ME,MNULL(2)
      COMMON/ASSE/NSYM,NKG,NKE,NDLE
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/VALP/NITER,NMDIAG,EPSLB,SHIFT,NSS,NSWM,TOLJAC,NVALP
      COMMON/ES/M,MR,MP,M1,M2,MDUMMY(8)
      COMMON/LOC/LCORG,LDLNC,LNEQ,LDIMP,LPRNG,LPREG,LLD,LLOCE,LCORE,LNE,
     1  LPRNE,LPREE,LDLE,LKE,LFE,LKGS,LKGD,LKSI,LFG,LRES,LDLG,LDUMMY(4)
      COMMON/TRVL/X1,X2,X3,I1,I2,I3,I4,I5,RDUMMY(515)
      COMMON VA(1)
      DIMENSION TBL(20)
      DATA ZERO/0.D0/
C+++     THIS IS COMMENTED OUT BECAUSE OF AN MS FORTRAN COMPILER
C+++     BUG WHICH WILL NOT INITIALIZE  $LARGE ARRAYS.  THIS ARRAY
C+++     IS NOW INITIALIZED BY A CALL TO A DUMMY SUBROUTINE
C+++     INITBL WHICH EXISTS SOLELY TO INITIALIZE THIS ARRAY
C
C     DATA TBL/'KGS ','KGD ','MGS ','MGD ','FG  ','KE  ','FE  ','DLE ',
C    1  'RES ','DLG ','P   ','LAMB','LAMI','R   ','PHI ','KSS ','MSS ',
C    1  'V1  ','VX  ','V2  '/
C
C        HERE IS THE CALL TO GET AROUND THE COMPILER BUG
      CALL INITBL(TBL,'VALP')
C
C+++     ALL OF THIS IS TO GET AROUND THE MICROSOFT
C+++     COMPILER BUG
C
C-----------------------------------------------------------------------
      IF(M1.EQ.0) M1=MR
      IF(M2.EQ.0) M2=ME
      READ(M1,1000) I1,I2,X1,X2,I3,I4,I5,X3
1000  FORMAT(2I5,2F10.0,3I5,1F10.0)
      IF(I1.NE.0) NVALP=I1
      IF(I2.NE.0) NITER=I2
      NSS=I3
      IF(I4.NE.0) NMDIAG=I4
      IF(I5.NE.0) NSWM=I5
      IF(X1.NE.ZERO) EPSLB=X1
      IF(X2.NE.ZERO) SHIFT=X2
      IF(X3.NE.ZERO) TOLJAC=X3
      IF(NSS.NE.0) GO TO 10
      NSS=MIN0(NVALP+8,2*NVALP)
      NSS=MIN0(NSS,NEQ)
```

```
10    CONTINUE
      WRITE(MP,2000) M,NVALP,NITER,NMDIAG,EPSLB,SHIFT,NSS,NSWM,TOLJAC
2000  FORMAT(//' SUBSPACE ITERATION (M=',I2,')'/' ',26('=')/
     1  15X,'NUMBER OF DESIRED EIGENVALUES              (NVALP)=',I12/
     2  15X,'MAX. NUMBER OF ITERATIONS PERMITTED        (NITER)=',I12/
     3  15X,'INDEX FOR DIAGONAL MATRIX                 (NMDIAG)=',I12/
     4  15X,'CONVERGENCE TOLERANCE ON EIGENVALUES       (EPSLB)=',E12.5/
     5  15X,'SHIFT                                      (SHIFT)=',E12.5/
     6  15X,'SUBSPACE DIMENSION                           (NSS)=',I12/
     7  15X,'MAX. NUMBER OF ITERATION IN JACOBI          (NSWM)=',I12/
     8  15X,'CONVERGENCE TOLERANCE IN JACOBI           (TOLJAC)=',1E12.5/)
      IF(NVALP.LE.NEQ.AND.NSS.LE.NEQ) GO TO 20
      WRITE(MP,2010)
2010  FORMAT(//'---ERROR--- NVALP OR NSS GREATER THAN NEQ',/,
     1        '---STOP EXECUTION---')
      GO TO 30
20    IF(LKGS.EQ.1) CALL ESPACE(NKG,1,TBL(1),LKGS)
      IF(LKGD.EQ.1) CALL ESPACE(NEQ,1,TBL(2),LKGD)
      CALL ESPACE(NKG,1,TBL(3),LMGS)
      CALL ESPACE(NEQ,1,TBL(4),LMGD)
      IF(LFG.EQ.1) CALL ESPACE(NEQ,1,TBL(5),LFG)
      IF(LKE.EQ.1) CALL ESPACE(NKE,1,TBL(6),LKE)
      IF(LFE.EQ.1) CALL ESPACE(NDLE,1,TBL(7),LFE)
      IF(LDLE.EQ.1) CALL ESPACE(NDLE,1,TBL(8),LDLE)
      IF(LRES.EQ.1) CALL ESPACE(NEQ,1,TBL(9),LRES)
      IF(LDLG.EQ.1) CALL ESPACE(NEQ,1,TBL(10),LDLG)
      CALL ESPACE(NEQ*NSS,1,TBL(11),LVEC)
      CALL ESPACE(NSS,1,TBL(12),LLAMB)
      CALL ESPACE(NSS,1,TBL(13),LLAM1)
      CALL ESPACE(NSS*(NSS+1)/2,1,TBL(16),LKSS)
      CALL ESPACE(NSS*(NSS+1)/2,1,TBL(17),LMSS)
      CALL ESPACE(NEQ,1,TBL(18),LV1)
      CALL ESPACE(NSS*NSS,1,TBL(19),LX)
      CALL EXVALP(VA(LLD),VA(LDIMP),VA(LLOCE),VA(LCORE),VA(LPRNE),
     1  VA(LPREE),VA(LNE),VA(LFE),VA(LKE),VA(LKGS),VA(LKGD),VA(LFG),
     2  VA(LCORG),VA(LDLNC),VA(LNEQ),VA(LRES),VA(LDLE),VA(LDLG),
     3  VA(LMGS),VA(LMGD),VA(LVEC),VA(LLAMB),VA(LLAM1),VA(LKSS),VA(LMSS)
     4  ,VA(LV1),VA(LX),NEQ,NSS)
30    RETURN
      END
```

```
      SUBROUTINE EXVALP(KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VFE,VKE,
     1  VKGS,VKGD,VFG,VCORG,KDLNC,KNEQ,VRES,VDLE,VDLG,VMGS,VMGD,
     2  VEC,VLAMB,VLAM1,VKSS,VMSS,V1,VX,NEQ,NSS)
C=======================================================================
C     TO EXECUTE BLOCK 'VALP'
C     TO COMPUTE EIGENVALUES AND EIGENVECTORS BY SUBSPACE
C     ITERATION
C     (IF  NVALP.EQ.1  INVERSE ITERATION METHOD)
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ASSE/NSYM,NKG,NKE,NDLE
      COMMON/VALP/NITER,NMDIAG,EPSLB,SHIFT,NSS1,NSWM,TOLJAC,NVALP
      COMMON/ES/M,MR,MP,MDUMMY(10)
      DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),VPRNE(*),VPREE(*),
     1  KNE(*),VFE(*),VKE(*),VKGS(*),VKGD(*),VFG(*),VCORG(*),KDLNC(*),
     2  KNEQ(*),VRES(*),VDLE(*),VDLG(*),VMGS(*),VMGD(*),VEC(NEQ,*),
     3  VLAMB(*),VLAM1(*),VKSS(*),VMSS(*),V1(*),VX(NSS,*)
      DATA ZERO/0.D0/,UN/1.0D0/,GRAND/1.0D32/
      ABS(X)=DABS(X)
C-----------------------------------------------------------------------
C
C------  PRELIMINARY COMPUTATIONS
C
C------  ASSEMBLE KG AND MG
      CALL ASKG(KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,VKGS,VKGD,
     1  VKGI,VFG,VDLE,VRES)
      CALL ASMG(KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,VMGS,
     1  VMGD,VMGS,VFG,VDLE,VRES)
C------  TRIANGULARIZE KG
      CALL SOL(VKGS,VKGD,VKGI,VFG,KLD,NEQ,MP,1,0,0,ENERG)
C------  LOAD VECTOR EQUAL TO DIAGONAL OF M
      CMAX=ZERO
      ICONT=0
      DO 10 ID=1,NEQ
      C=GRAND
C------ CHECK FOR ZERO DIAGONAL TERM IN VMGD
      IF(VMGD(ID).EQ.ZERO) GO TO 5
      ICONT=ICONT+1
      C=VKGD(ID)/VMGD(ID)
5     V1(ID)=C
      IF(C.GT.CMAX) CMAX=C
      VEC(ID,1)=VMGD(ID)
      DO 10 JS=2,NSS
10    VEC(ID,JS)=ZERO
C------ CHECK IF SUBSPACE DIMENSION IS EQUAL TO MASS D.O.F.
      IF(ICONT.LT.NSS) GO TO 250
C------  UNIT LOAD VECTORS CORRESPONDING TO MIN. OF
C         K(I,I)/M(I,I)
      DO 30 JS=2,NSS
      C=CMAX
```

```fortran
      DO 20 ID=1,NEQ
      IF(V1(ID).GT.C) GO TO 20
      C=V1(ID)
      II=ID
20    CONTINUE
      V1(II)=CMAX
      VEC(II,JS)=UN
30    VLAMB(JS)=UN
      VLAMB(1)=UN
      IF(NVALP.EQ.1) NSS=1
C------- INVERSE ITERATION IF NVALP=1
C------- START  ITERATIONS LOOP
C
      ITERM=0
      ITMAX=NITER+1
      DO 200 ITER=1,ITMAX
C------- COMPUTE RITZ VECTORS
      II0=0
      DO 80 JS=1,NSS
      II0=II0+JS
      DO 40 ID=1,NEQ
40    V1(ID)=VEC(ID,JS)
      CALL SOL(VKGS,VKGD,VKGI,V1,KLD,NEQ,MP,0,1,0,ENERG)
C------- CALCULATE THE PROJECTION OF K
      II=II0
      DO 60 IS=JS,NSS
      C=ZERO
      DO 50 ID=1,NEQ
50    C=C+V1(ID)*VEC(ID,IS)
      VKSS(II)=C
60    II=II+IS
      DO 70 ID=1,NEQ
70    VEC(ID,JS)=V1(ID)
80    CONTINUE
C------- CALCULATE THE PROJECTION OF M
      II0=0
      DO 120 JS=1,NSS
      II0=II0+JS
      DO 85 ID=1,NEQ
85    V1(ID)=ZERO
      CALL MULKU(VMGS,VMGD,VMGS,KLD,VEC(1,JS),NEQ,0,V1)
      II=II0
      DO 100 IS=JS,NSS
      C=ZERO
      DO 90 ID=1,NEQ
90    C=C+V1(ID)*VEC(ID,IS)
      IF(ITERM.GT.0) GO TO 120
      VMSS(II)=C
100   II=II+IS
      DO 110 ID=1,NEQ
110   VEC(ID,JS)=V1(ID)
```

```
120   CONTINUE
      IF (NSS.GT.1) GO TO 125
      VLAM1(1)=VKSS(1)/VMSS(1)
      GO TO 165
C------- CALCULATE EIGENVALUES IN THE SUBSPACE
125   CALL JACOBI(VKSS,VMSS,NSS,NSWM,TOLJAC,V1,VLAM1,VX)
C------- NEW LOAD VECTOR
      DO 160 ID=1,NEQ
      DO 130 JS=1,NSS
130   V1(JS)=VEC(ID,JS)
      DO 150 JS=1,NSS
      C=ZERO
      DO 140 IS=1,NSS
140   C=C+V1(IS)*VX(IS,JS)
150   VEC(ID,JS)=C
160   CONTINUE
165   CONTINUE
C------- PRINT THE ITERATION VALUES
      IF(M.LT.1) GO TO 180
      WRITE(MP,2000) ITER
2000  FORMAT(//' . . . . . . . ITERATION ',I5/)
      DO 170 IS=1,NSS
      WRITE(MP,2010) IS,VLAM1(IS)
2010  FORMAT(/' EIGENVALUE NO. ',I5,' =',E12.5//' EIGENVECTOR:')
170   CALL PRSOL(KDLNC,VCORG,VDIMP,KNEQ,VEC(1,IS))
C------- CHECK FOR CONVERGENCE
180   IF(ITERM.GT.0) GO TO 210
      C=ZERO
      IEX=0
      DO 190 IS=1,NSS
      C1=ABS((VLAM1(IS)-VLAMB(IS))/VLAMB(IS))
      IF(C1.GT.C) C=C1
      IF(C1.LE.EPSLB) IEX=IEX+1
190   CONTINUE
      WRITE(MP,2015) ITER,C,IEX
2015   FORMAT(' ITERATION ',I4,' MAX. ERROR=',E9.1,' EXACT EIGENVALUES:'
     1,I4)
      IF(IEX.GE.NVALP) ITERM=1
C------- NON CONVERGENCE
      IF(ITER.LT.NITER.OR.ITERM.EQ.1) GO TO 195
      WRITE(MP,2020) NITER
2020  FORMAT(' ** NON CONVERGENCE AFTER ',I5,' ITERATIONS')
      ITERM=1
C------- SAVE THE EIGENVALUES
195   DO 200 IS=1,NSS
200   VLAMB(IS)=VLAM1(IS)
C
C------- RESULT
C
C------- ARRANGE EIGENVALUES IN ASCENDING ORDER
210   IS1=NSS-1
```

```
      IF (IS1.EQ.0) GO TO 235
      DO 230 IS=1,IS1
      I1=IS+1
      C=VLAMB(IS)
      II=IS
      DO 220 JS=I1,NSS
      IF(C.LT.VLAMB(JS)) GO TO 220
      C=VLAMB(JS)
      II=JS
220   CONTINUE
      VLAMB(II)=VLAMB(IS)
      VLAMB(IS)=C
      DO 230 ID=1,NEQ
      C=VEC(ID,IS)
      VEC(ID,IS)=VEC(ID,II)
230   VEC(ID,II)=C
C------- PRINT RESULT
      WRITE(MP,2030) ITER
2030  FORMAT(/' . . . . CONVERGENCE IN',I4,' ITERATIONS'/)
235   CONTINUE
      DO 240 IS=1,NVALP
      WRITE(MP,2010) IS,VLAMB(IS)
240   CALL PRSOL(KDLNC,VCORG,VDIMP,KNEQ,VEC(1,IS))
      GO TO 260
250   CONTINUE
      WRITE(MP,2040)
2040  FORMAT('  ** NSS IS LARGER THAN MASS D.O.F.')
260   RETURN
      END
```

```
      SUBROUTINE ASMG(KLD,VDIMP,KLOCE,VCORE,VPRNE,VPREE,KNE,VKE,VFE,
     1  VKGS,VKGD,VKGI,VFG,VDLE,VRES)
C=====================================================================
C     TO ASSEMBLE THE GLOBAL MASS MATRIX (ELEMENT FUNCTION 5)
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ELEM/NELT,NNEL,NTPE,NGRE,ME,NIDENT,MNULL
      COMMON/ASSE/NSYM,MFILLR(3)
      COMMON/RESO/NEQ,NFILLR(2)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,IPG
     1  ,ICOD,NULL(3)
      COMMON/ES/M,MR,MP,M1,M2,MDUMMY(8)
      DIMENSION KLD(*),VDIMP(*),KLOCE(*),VCORE(*),VPRNE(*),VPREE(*),
     1  KNE(*),VKE(*),VFE(*),VKGS(*),VKGD(*),VKGI(*),VFG(*),VDLE(*),
     2  VRES(*),KEB(1)
C---------------------------------------------------------------------
C------- REWIND ELEMENT FILE (M2)
      REWIND M2
C------- LOOP OVER THE ELEMENTS
      DO 30 IE=1,NELT
C------- SKIP COMPUTATIONS IF IDENTICAL ELEMENTS
      IF(NIDENT.EQ.1.AND.IE.GT.1) GO TO 20
C------- READ AN ELEMENT
      CALL RDELEM(M2,KLOCE,VCORE,VPRNE,VPREE,KNE)
C------- EVALUATE INTERPOLATION FUNCTIONS IF REQUIRED
      IF(ITPE.EQ.ITPE1) GO TO 10
      ICOD=2
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
10    ICOD=5
      CALL ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C------- PRINT ELEMENT MATRIX
      IF(M.LT.2) GO TO 20
      IF(NSYM.EQ.0) IKE=IDLE*(IDLE+1)/2
      IF(NSYM.EQ.1) IKE=IDLE*IDLE
      WRITE(MP,2000) IEL,(VKE(I),I=1,IKE)
2000  FORMAT(/' MATRIX (ME) , ELEMENT:',I5/(10X,10E12.5))
C------- ASSEMBLE
20    CALL ASSEL(1,0,IDLE,NSYM,KLOCE,KLD,VKE,VFE,VKGS,VKGD,VKGI,VFG)
30    ITPE1=ITPE
      RETURN
      END
```

```
      SUBROUTINE JACOBI(VK,VM,N,NCYM,EPS,VALPO,VALP,VECT)
C=======================================================================
C     TO SOLVE THE EIGENPROBLEM K-LAMBDA.M BY THE GENERALIZED
C     JACOBI METHOD
C       INPUT
C         VK        MATRIX K (UPPER TRIANGLE BY DESCENDING
C                   COLUMNS)
C         VM        MATRIX M (UPPER TRIANGLE BY DESCENDING
C                   COLUMNS)
C         N         ORDER OF MATRICES K AND M
C         NCYM      MAXIMUM NUMBER OF SWEEPS ALLOWED (15)
C         EPS       CONVERGENCE TOLERANCE (1.D-12)
C       WORKSPACE
C         VALPO     WORKING VECTOR (DIMENSION N)
C       OUTPUT
C         VALP      EIGENVALUES
C         VECT      EIGENVECTORS
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/ES/M,MR,MP,MDUMMY(10)
      DIMENSION VK(*),VM(*),VALPO(N),VALP(N),VECT(N,N)
      DATA EPSDO/1.D-4/,ZERO/0.D0/,UN/1.D0/,DEUX/2.D0/,QUATR/4.D0/
      SQRT(X)=DSQRT(X)
      ABS(X)=DABS(X)
      EPS2=EPS*EPS
      ITR=0
C------ VERIFY IF DIAGONAL TERMS ARE POSITIVE
C         AND INITIALIZE EIGENVALUES
      II=0
      DO 20 I=1,N
      II=II+I
      IF(VK(II).GT.ZERO.AND.VM(II).GT.ZERO) GO TO 10
      WRITE(MP,2000) I
2000  FORMAT(' ** ERROR, NEGATIVE DIAGONAL TERM IN JACOBI, ROW ',
     1  I5)
      STOP
10    VALP(I)=VK(II)/VM(II)
20    VALPO(I)=VALP(I)
C------ INITIALIZE EIGENVECTORS
      DO 40 I=1,N
      DO 30 J=1,N
30    VECT(I,J)=ZERO
40    VECT(I,I)=UN
C------- FOR EACH SWEEP
      DO 250 IC=1,NCYM
C------- DYNAMIC TOLERANCE
      EPSD=EPSDO**IC
C-------- SWEEP ROWWISE OVER UPPER TRIANGLE
      IMAX=N-1
       II=0
```

```
      DO 180 I=1,IMAX
      IO=II+1
      II=II+I
      IP1=I+1
      IJ=II+I
      JJ=II
      DO 180 J=IP1,N
      JP1=J+1
      JM1=J-1
      JO=JJ+1
      JJ=JJ+J
      J3=JJ-1
C------  COMPUTE COUPLING FACTORS
      FK=(VK(IJ)*VK(IJ))/(VK(II)*VK(JJ))
      FM=(VM(IJ)*VM(IJ))/(VM(II)*VM(JJ))
      IF(FK.LT.EPSD.AND.FM.LT.EPSD) GO TO 180
C------  COMPUTE THE TRANSFORMATION COEFFICIENTS
      ITR=ITR+1
      C1=VK(II)*VM(IJ)-VM(II)*VK(IJ)
      C2=VK(JJ)*VM(IJ)-VM(JJ)*VK(IJ)
      C3=VK(II)*VM(JJ)-VM(II)*VK(JJ)
      DET=(C3*C3/QUATR)+(C1*C2)
      IF(DET.GE.ZERO) GO TO 50
      WRITE(MP,2005) I,J
2005  FORMAT(' **ERROR, SINGULAR JACOBI TRANSFORMATION I=',I5,
     1   ' J=',I5)
      STOP
50    DET=SQRT(DET)
      D1=C3/DEUX+DET
      D2=C3/DEUX-DET
      D=D1
      IF(ABS(D2).GT.ABS(D1))D=D2
      IF(D.EQ.ZERO) GO TO 60
      A=C2/D
      B=-C1/D
      GO TO 65
60    A=ZERO
      B=-VK(IJ)/VK(JJ)
C-------  MODIFY COLUMNS OF K AND M
65    IF(I.EQ.1) GO TO 80
      IK=IO
      J1=IJ-1
      DO 70 JK=JO,J1
      C1=VK(IK)
      C2=VK(JK)
      VK(IK)=C1+B*C2
      VK(JK)=C2+A*C1
      C1=VM(IK)
      C2=VM(JK)
      VM(IK)=C1+B*C2
      VM(JK)=C2+A*C1
```

```
70    IK=IK+1
80    IF(I.EQ.JM1) GO TO 100
      IK=II+I
      J2=IJ+1
      IM=I
      DO 90 JK=J2,J3
      C1=VK(IK)
      C2=VK(JK)
      VK(IK)=C1+B*C2
      VK(JK)=C2+A*C1
      C1=VM(IK)
      C2=VM(JK)
      VM(IK)=C1+B*C2
      VM(JK)=C2+A*C1
      IM=IM+1
90    IK=IK+IM
100   IF(J.EQ.N) GO TO 120
      IK=IJ+J
      JK=JJ+J
      IM=J
      DO 110 JJK=JP1,N
      C1=VK(IK)
      C2=VK(JK)
      VK(IK)=C1+B*C2
      VK(JK)=C2+A*C1
      C1=VM(IK)
      C2=VM(JK)
      VM(IK)=C1+B*C2
      VM(JK)=C2+A*C1
      IM=IM+1
      IK=IK+IM
110   JK=JK+IM
120   C1=VK(II)
      C2=VK(IJ)
      C3=VK(JJ)
      B2=B*B
      BB=DEUX*B
      A2=A*A
      AA=DEUX*A
      VK(II)=C1+BB*C2+B2*C3
      VK(IJ)=ZERO
      VK(JJ)=C3+AA*C2+A2*C1
      C1=VM(II)
      C2=VM(IJ)
      C3=VM(JJ)
      VM(II)=C1+BB*C2+B2*C3
      VM(IJ)=ZERO
      VM(JJ)=C3+AA*C2+A2*C1
C-------  UPDATE EIGENVECTORS
      DO 170 IJ1=1,N
      C1=VECT(IJ1,I)
```

```
      C2=VECT(IJ1,J)
      VECT(IJ1,I)=C1+B*C2
170   VECT(IJ1,J)=C2+A*C1
180   IJ=IJ+J
C-------  UPDATE EIGENVALUES
      II=0
      DO 190 I=1,N
      II=II+I
      IF(VK(II).GT.ZERO.AND.VM(II).GT.ZERO) GO TO 190
      WRITE(MP,2000) I
      STOP
190   VALP(I)=VK(II)/VM(II)
      IF(M.GT.1) WRITE(MP,2010)IC,(VALP(I),I=1,N)
2010  FORMAT(/' EIGENVALUES, SWEEP ',I4/(1X,10E12.5))
C-------  CHECK FOR CONVERGENCE OF EIGENVALUES
      DO 200 I=1,N
      IF(ABS(VALP(I)-VALPO(I)).GT.(EPS*VALPO(I))) GO TO 230
200   CONTINUE
C------  CHECK FOR CONVERGENCE ON DIAGONAL TERMS
      JJ=1
      DO 210 J=2,N
      JJ=JJ+J
      JM1=J-1
      II=0
      DO 210 I=1,JM1
      II=II+I
      IJ=JJ-J+I
      FK=VK(IJ)*VK(IJ)/(VK(II)*VK(JJ))
      FM=VM(IJ)*VM(IJ)/(VM(II)*VM(JJ))
      IF(FK.GT.EPS2.OR.FM.GT.EPS2) GO TO 230
210   CONTINUE
C------  NORMALIZE EIGENVECTORS
      JJ=0
      DO 220 J=1,N
      JJ=JJ+J
      C1=SQRT(VM(JJ))
      DO 220 I=1,N
220   VECT(I,J)=VECT(I,J)/C1
C-------  ACHIEVED CONVERGENCE
      IF(M.GT.0) WRITE(MP,2020) IC,ITR
2020  FORMAT(15X,'CONVERGENCE IN ',I4,' SWEEPS AND ',I5,' TRANSFORMATION
     1S IN JACOBI')
      RETURN
C------  TRANSFER VALP INTO VALPO
230   DO 240 I=1,N
240   VALPO(I)=VALP(I)
250   CONTINUE
C-------  FAIL TO CONVERGE
      WRITE(MP,2030) NCYM
2030  FORMAT(' ** ERROR, CONVERGENCE FAILURE IN JACOBI IN',I4,' SWEEPS')
      STOP
      END
```

```
$LARGE: VKSI1
$LARGE: KEXP1
$LARGE: VKSI2
$LARGE: KEXP2
$LARGE: VKSI3
$LARGE: KEXP3
$LARGE: VKSI
$LARGE: KEXP
$LARGE: INDIC
$LARGE: G
$LARGE: P
$LARGE: TBL
$LARGE: WGT
$LARGE: PSIT
$LARGE: ETAT
$LARGE: INTNUM
$LARGE: NINTV
$LARGE: PS
$LARGE: ET
$LARGE: IPGKED
      SUBROUTINE DUMMY(MICROSOFT,BUG,KILLER)
C=====================================================================
C       I REFER TO THE FOLLOWING SUBROUTINES, WHOSE NAMES BEGIN
C       WITH "INIT," AS DUMMY SUBROUTINES, BECAUSE THEY ARE
C       NEEDED TO INITIALIZE THE ARRAYS WHICH ARE PASSED AS
C       CALLING PARAMETERS.  THE ARRAYS CANNOT BE INITIALIZED WITH
C       DATA STATEMENTS IN A DIRECT FASHION BECAUSE THERE IS A
C       BUG IN THE MICROSOFT FORTRAN COMPILER V3.2, WHICH DOES NOT
C       INITIALIZE REAL ARRAYS CORRECTLY IF THEY HAVE BEEN IDENT-
C       IFIED AS $LARGE ARRAYS.  IT DOES NOT SEEM TO MATTER WHETHER
C       THEY ARE DECLARED LARGE USING THE "GENERIC" $LARGE WITHOUT
C       SPECIFIC ARGUMENTS, OR WHETHER THEY HAVE BEEN DECLARED
C       SPECIFICALLY AS IN THE METACOMMANDS PRECEEDING THIS ROUTINE
C
C       IF THE BELOW ROUTINES ARE DUMMY ROUTINES; THIS ONE HAS GOT
C       TO BE CALLED AN IDIOT ROUTINE.  THIS ROUTINE EXISTS BECAUSE
C       THE DATA STATEMENTS FOR REAL ARRAYS WILL NOT COMPILE CORRECTLY
C       IF THEY ARE IN THE FIRST SUBROUTINE IN A COMPILAND.  THIS
C       SUBROUTINE PROVIDES A PAD TO FOOL THE COMPILER.  WITHOUT
C       THIS ROUTINE, THE ONE IMMEDIATELY FOLLOWING WILL NOT COMPILE;
C       WITH THIS ROUTINE IT DOES.
C
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION BUG(20)
      BUG(1) = 0.D0
      RETURN
      END
```

```
      SUBROUTINE INITN1(VKSI1,KEXP1,VKSI2,KEXP2,VKSI3,KEXP3)
C=====================================================================
C        THIS SUBROUTINE EXISTS SOLELY TO GET AROUND A MICROSOFT
C        COMPILER BUG.  ITS PURPOSE IS TO INITIALIZE THE ARRAYS
C        PASSED AS ARGUMENTS.  THE DUMMY ARRAYS VKSI11, VKSI22,
C        VKSI33, KEXP11, KEXP22, AND KEXP33 HAVE BEEN GIVEN THE
C        ATTRIBUTE $NOTLARGE, AND WILL BE INITIALIZED PROPERLY
C        BY THE COMPILER.  THE $NOTLARGE ATTRIBUTE IS ASSIGNED
C        BY DEFAULT SINCE THEIR DIMENSIONS DO NOT EXCEED 64K BYTES
C        OF STORAGE.
C
C        THIS SUBROUTINE IS CALLED BY SUBROUTINE XI01 WHICH IS
C        USED FOR ELEMENT TYPE 1
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION VKSI1 (3),KEXP1 (3),VKSI2 (16),KEXP2 (16),VKSI3 (60),
     1  KEXP3(60)
      DIMENSION VKSI11(3),KEXP11(3),VKSI22(16),KEXP22(16),VKSI33(60),
     1  KEXP33(60)
C
C   CHARACTERISTICS FOR 1,2 AND 3 DIMENSIONAL REFERENCE ELEMENTS
C
C             HERE IS THE DUMMY ARRAY INITIALIZATION
C
      DATA VKSI11/-1.D0,0.D0,1.D0/
      DATA KEXP11/0,1,2/
      DATA VKSI22/-1.D0,-1.D0, +0.D0,-1.D0, +1.D0,-1.D0, +1.D0,+0.D0,
     1           +1.D0,+1.D0, +0.D0,+1.D0, -1.D0,+1.D0, -1.D0,+0.D0/
      DATA KEXP22/0,0, 1,0, 0,1, 2,0, 1,1, 0,2, 2,1, 1,2/
      DATA VKSI33/-1.D0,-1.D0,-1.D0, +0.D0,-1.D0,-1.D0,
     1           +1.D0,-1.D0,-1.D0, +1.D0,+0.D0,-1.D0,
     2           +1.D0,+1.D0,-1.D0, +0.D0,+1.D0,-1.D0,
     3           -1.D0,+1.D0,-1.D0, -1.D0,+0.D0,-1.D0,
     4           -1.D0,-1.D0,+0.D0, +1.D0,-1.D0,+0.D0,
     5           +1.D0,+1.D0,+0.D0, -1.D0,+1.D0,+0.D0,
     6           -1.D0,-1.D0,+1.D0, +0.D0,-1.D0,+1.D0,
     7           +1.D0,-1.D0,+1.D0, +1.D0,+0.D0,+1.D0,
     8           +1.D0,+1.D0,+1.D0, +0.D0,+1.D0,+1.D0,
     9           -1.D0,+1.D0,+1.D0, -1.D0,+0.D0,+1.D0/
      DATA KEXP33/0,0,0, 1,0,0, 0,1,0, 0,0,1, 1,1,1,
     1  1,1,0, 0,1,1, 1,0,1, 2,0,0, 0,2,0, 0,0,2,
     2  2,1,0, 2,0,1, 2,1,1, 1,2,0, 0,2,1, 1,2,1,
     3  1,0,2, 0,1,2, 1,1,2/
C
C          INITIALIZE THE REAL ARRAYS
C
      DO 10 I = 1,3
        VKSI1(I) = VKSI11(I)
        KEXP1(I) = KEXP11(I)
10    CONTINUE
```

```
      DO 20 I = 1,16
        VKSI2(I) = VKSI22(I)
        KEXP2(I) = KEXP22(I)
20    CONTINUE
      DO 30 I = 1,60
        VKSI3(I) = VKSI33(I)
        KEXP3(I) = KEXP33(I)
30    CONTINUE
      RETURN
      END


      SUBROUTINE INITN2(VKSI,KEXP)
C=======================================================================
C        THIS SUBROUTINE EXISTS SOLELY TO GET AROUND A MICROSOFT
C        COMPILER BUG.  ITS PURPOSE IS TO INITIALIZE THE ARRAYS
C        PASSED AS ARGUMENTS.  THE DUMMY ARRAYS VKSII AND KEXPP HAVE
C        BEEN GIVEN THE ATTRIBUTE $NOTLARGE, AND WILL BE INITIALIZED
C        PROPERLY BY THE COMPILER.  THE $NOTLARGE ATTRIBUTE IS ASSIGNED
C        BY DEFAULT SINCE THEIR DIMENSIONS DO NOT EXCEED 64K BYTES
C        OF STORAGE.
C
C        THIS SUBROUTINE IS CALLED BY SUBROUTINE NI02 WHICH IS
C        USED BY ELEMENT TYPE 2
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
C
C......  INFORMATION RELATED TO THE 8 NODED REFERENCE SQUARE ELEMENT
C          (INEL.EQ.8  NDIM.EQ.2)
C     DIMENSION VKSI(NDIM*INEL),KEXP(NDIM*INEL),KDER(NDIM)
      DIMENSION VKSI (       16),KEXP (       16)
      DIMENSION VKSII(       16),KEXPP(       16)
C
C            INTITIALIZE THE DUMMY ARRAYS
C
C         NODAL COORDINATES OF THE REFERENCE ELEMENT
      DATA VKSII/-1.D0,-1.D0,  +0.D0,-1.D0,  +1.D0,-1.D0,  +1.D0,+0.D0,
     1           +1.D0,+1.D0,  +0.D0,+1.D0,  -1.D0,+1.D0,  -1.D0,+0.D0/
C         MONOMIAL EXPONENTS OF THE POLYNOMIAL BASIS, MAX-DEGREE
      DATA KEXPP/0,0,  1,0,  0,1,  2,0,  1,1,  0,2,  2,1,  1,2/
C
C            INITIALIZE THE REAL ARRAYS
C
      DO 10 I = 1,16
        KEXP(I) = KEXPP(I)
        VKSI(I) = VKSII(I)
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE INITN3(VKSI,KEXP)
C=====================================================================
C        THIS SUBROUTINE EXISTS SOLELY TO GET AROUND A MICROSOFT
C        COMPILER BUG.  ITS PURPOSE IS TO INITIALIZE THE ARRAYS
C        PASSED AS ARGUMENTS.  THE DUMMY ARRAYS VKSII AND KEXPP HAVE
C        BEEN GIVEN THE ATTRIBUTE $NOTLARGE, AND WILL BE INITIALIZED
C        PROPERLY BY THE COMPILER.  THE $NOTLARGE ATTRIBUTE IS ASSIGNED
C        BY DEFAULT SINCE THEIR DIMENSIONS DO NOT EXCEED 64K BYTES
C        OF STORAGE.
C
C        THIS SUBROUTINE IS CALLED BY SUBROUTINE NI03 WHICH IS
C        USED BY ELEMENT TYPE 3
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
C
C.......  INFORMATIONS CARACTERISTIQUES DU TRIANGLE A 6 NOEUDS
C          (INEL.EQ.6  NDIM.EQ.2)
C     DIMENSION VKSI(NDIM*INEL),KEXP(NDIM*INEL)
      DIMENSION VKSI (       12),KEXP (       12)
      DIMENSION VKSII(       12),KEXPP(       12)
C
C        THIS IS THE DUMMY ARRAY INITIALIZATION
C
C         COORDONNEES DES NOEUDS DE L'ELEMENT DE REFERENCE
      DATA VKSII/0.D0,0.D0,0.5D0,0.D0,1.D0,0.D0,0.5D0,0.5D0,0.D0,1.D0,
     1           0.D0,0.5D0/
C         EXPOSANTS DES MONOMES DE LA BASE POLYNOMIALE,DEGRE MAX.
      DATA KEXPP/0,0, 1,0, 0,1, 2,0, 1,1, 0,2/
C
C        INITIALIZE THE REAL ARRAYS
C
      DO 10 I = 1,12
        VKSI(I) = VKSII(I)
        KEXP(I) = KEXPP(I)
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE INITS6(WGT,PSIT,ETAT,INTNUM,NINTV)
C=======================================================================
C        THIS SUBROUTINE EXISTS SOLELY TO GET AROUND A MICROSOFT
C        COMPILER BUG.  ITS PURPOSE IS TO INITIALIZE THE ARRAYS
C        PASSED AS ARGUMENTS.  THE DUMMY ARRAYS WGTT,PSITT,
C        ETATT, INTNUU, AND NINTVV HAVE BEEN GIVEN THE
C        ATTRIBUTE $NOTLARGE, AND WILL BE INITIALIZED PROPERLY
C        BY THE COMPILER.  THE $NOTLARGE ATTRIBUTE IS ASSIGNED
C        BY DEFAULT SINCE THEIR DIMENSIONS DO NOT EXCEED 64K BYTES
C        OF STORAGE.
C
C        THIS SUBROUTINE IS CALLED BY SUBROUTINE STF06 WHICH IS
C        USED FOR ELEMENT TYPE 6
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION WGT (*),PSIT (*),ETAT (*),INTNUM(*),NINTV (*)
      DIMENSION WGTT(7),PSITT(7),ETATT(7),INTNUU(5),NINTVV(5)
C
C             HERE IS THE DUMMY ARRAY INITIALIZATION
C
      DATA PSITT/ 0.333333333333D0,
     *          0.166666666667D0,0.166666666667D0,0.666666666667D0,
     *          0.5D0           ,0.5D0           ,0.0D0/
      DATA ETATT/ 0.333333333333D0,
     *          0.166666666667D0,0.666666666667D0,0.166666666667D0,
     *          0.0D0           ,0.5D0           ,0.5D0/
      DATA WGTT/ 1.0D0,
     *         0.333333333333D0,0.333333333333D0,0.333333333333D0,
     *         0.333333333333D0,0.333333333333D0,0.333333333333D0/
      DATA INTNUU / 0,1,4,7,11/
      DATA NINTVV / 1,3,3,4, 7/
C
C          INITIALIZE THE REAL ARRAYS
C
      DO 10 I = 1,5
        INTNUM(I) = INTNUU(I)
        NINTV(I) = NINTVV(I)
10    CONTINUE
      DO 20 I = 1,7
        WGT(I) = WGTT(I)
        PSIT(I) = PSITT(I)
        ETAT(I) = ETATT(I)
20    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE INITR6(PS,ET)
C==================================================================
C        THIS SUBROUTINE EXISTS SOLELY TO GET AROUND A MICROSOFT
C        COMPILER BUG.  ITS PURPOSE IS TO INITIALIZE THE ARRAYS
C        PASSED AS ARGUMENTS.  THE DUMMY ARRAYS PSS AND ETT HAVE
C        BEEN GIVEN THE ATTRIBUTE $NOTLARGE, AND WILL BE INITIALIZED
C        PROPERLY BY THE COMPILER.  THE $NOTLARGE ATTRIBUTE IS ASSIGNED
C        BY DEFAULT SINCE THEIR DIMENSIONS DO NOT EXCEED 64K BYTES
C        OF STORAGE.
C
C        THIS SUBROUTINE IS CALLED BY SUBROUTINE STRS6 WHICH IS
C        USED FOR ELEMENT TYPE 6
C==================================================================
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION PS (*),ET (*)
      DIMENSION PSS(6),ETT(6)
C
C             HERE IS THE DUMMY ARRAY INITIALIZATION
C
      DATA PSS /0.,1.,0.,0.5,0.5,0./
      DATA ETT /0.,0.,1.,0.,0.5,0.5/
C
C           INITIALIZE THE REAL ARRAYS
C
      DO 10 I = 1,6
        PS(I) = PSS(I)
        ET(I) = ETT(I)
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE INITN7(VKSI,KEXP)
C=================================================================
C        THIS SUBROUTINE EXISTS SOLELY TO GET AROUND A MICROSOFT
C        COMPILER BUG.  ITS PURPOSE IS TO INITIALIZE THE ARRAYS
C        PASSED AS ARGUMENTS.  THE DUMMY ARRAYS VKSII AND KEXPP HAVE
C        BEEN GIVEN THE ATTRIBUTE $NOTLARGE, AND WILL BE INITIALIZED
C        PROPERLY BY THE COMPILER.  THE $NOTLARGE ATTRIBUTE IS ASSIGNED
C        BY DEFAULT SINCE THEIR DIMENSIONS DO NOT EXCEED 64K BYTES
C        OF STORAGE.
C
C        THIS SUBROUTINE IS CALLED BY SUBROUTINE NI07 WHICH IS
C        USED BY ELEMENT TYPE 7
C=================================================================
      IMPLICIT REAL*8(A-H,O-Z)
C
C.......  INFORMATIONS LIEES A L'ELEMENT DE REFERENCE CARRE A 20 NOEUDS
C          (INEL.EQ.20  NDIM.EQ.3)
C     DIMENSION VKSI(NDIM*INEL),KEXP(NDIM*INEL),KDER(NDIM)
      DIMENSION VKSI(       60),KEXP(       60)
      DIMENSION VKSII(      60),KEXPP(      60)
C
C        INITIALIZE THE DUMMY ARRAYS
C
      DATA VKSII/
     1  -1.D0,-1.D0,-1.D0,  +0.D0,-1.D0,-1.D0,  +1.D0,-1.D0,-1.D0,
     2  +1.D0,+0.D0,-1.D0,
     3  +1.D0,+1.D0,-1.D0,  +0.D0,+1.D0,-1.D0,  -1.D0,+1.D0,-1.D0,
     4  -1.D0,+0.D0,-1.D0,
     5  -1.D0,-1.D0,+0.D0,  +1.D0,-1.D0,+0.D0,  +1.D0,+1.D0,+0.D0,
     6  -1.D0,1.D0,+0.D0,
     7  -1.D0,-1.D0,+1.D0,  +0.D0,-1.D0,+1.D0,  +1.D0,-1.D0,+1.D0,
     8  +1.D0,+0.D0,+1.D0,
     9  +1.D0,+1.D0,+1.D0,  +0.D0,+1.D0,+1.D0,  -1.D0,+1.D0,+1.D0,
     A  -1.D0,+0.D0,+1./
C         EXPOSANTS DES MONOMES DE LA BASE POLYNOMIALE,DEGRE MAX.
      DATA KEXPP/0,0,0, 1,0,0, 0,1,0, 0,0,1, 2,0,0, 0,2,0, 0,0,2,
     1           1,1,0, 0,1,1, 1,0,1, 2,1,0, 2,0,1, 1,2,0, 0,2,1,
     2           1,0,2, 0,1,2, 1,1,1, 2,1,1, 1,2,1, 1,1,2/
C
C           INITIALIZE THE REAL ARRAYS
C
      DO 10 I = 1,60
        KEXP(I) = KEXPP(I)
        VKSI(I) = VKSII(I)
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE INITGA(INDIC,G,P)
C=====================================================================
C        THIS SUBROUTINE EXISTS SOLELY TO GET AROUND A MICROSOFT
C        COMPILER BUG.  ITS PURPOSE IS TO INITIALIZE THE ARRAYS
C        PASSED AS ARGUMENTS.  THE DUMMY ARRAYS INDICC, GG, AND
C        PP HAVE BEEN GIVEN THE ATTRIBUTE $NOTLARGE, AND WILL BE
C        INITIALIZED PROPERLY BY THE COMPILER.  THE $NOTLARGE
C        ATTRIBUTE IS ASSIGNED BY DEFAULT SINCE THEIR DIMENSIONS
C        DO NOT EXCEED 64K BYTES OF STORAGE.
C
C        THIS SUBROUTINE IS CALLED BY SUBROUTINE GAUSS.
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION INDIC (4),G (10),P (10)
      DIMENSION INDICC(4),GG(10),PP(10)
C
C             HERE IS THE DUMMY ARRAY INITIALIZATION
C
      DATA INDICC/1,2,4,7/
      DATA GG/0.0D0,-.5773502691896260D0,.5773502691896260D0,
     1       -.7745966692414830D0,0.0D0,.7745966692414830D0,
     2       -.8611363115940500D0,-.3399810435848600D0,
     3       .3399810435848600D0,.8611363115940500D0/
      DATA PP/2.0D0,1.0D0,1.0D0,
     1      0.5555555555555556D0,0.8888888888888889D0,0.5555555555555556D0,
     2       .3478548451374500D0,.6521451548625500D0,
     3       .6521451548625500D0,.3478548451374500D0/
C
C             INITIALIZE THE REAL ARRAYS
C
      DO 10 I = 1, 4
        INDIC(I) = INDICC(I)
10    CONTINUE
      DO 20 I = 1,10
        G(I) = GG(I)
        P(I) = PP(I)
20    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE INITBL(TBL,WHO)
C=====================================================================
C        THIS SUBROUTINE EXISTS SOLELY TO GET AROUND A MICROSOFT
C        COMPILER BUG.  ITS PURPOSE IS TO INITIALIZE THE ARRAY TBL
C        PASSED AS AN ARGUMENT.  THE DUMMY ARRAYS TBLX HAVE BEEN GIVEN
C        THE ATTRIBUTE $NOTLARGE, AND WILL BE INITIALIZED PROPERLY
C        BY THE COMPILER.  THE $NOTLARGE ATTRIBUTE IS ASSIGNED BY
C        DEFAULT SINCE THE DIMENSION DOES NOT EXCEED 64K BYTES OF
C        STORAGE.  TBL CONTAINS POINTERS INTO THE WORKING ARRAY.
C
C        THIS SUBROUTINE IS CALLED BY SUBROUTINE BLLIND, BLNLIN,
C        BLTEMP, BLVALP, BLCOOR, BLCOND, BLPREL, BLELEM, BLSOLR,
C        AND BLLINM.
C=====================================================================
      CHARACTER*4 TBL,WHO,CALLER(10),TBL1(10),TBL2(10),TBL3(13),
     *                             TBL4(20),TBL5( 2),TBL6( 2),
     *                             TBL7( 2),TBL8( 6),TBL9( 8),TBL10(8)
      DIMENSION TBL (*)
      DATA CALLER/'LIND','NLIN','TEMP','VALP','COOR','COND','PREL',
     *            'ELEM','SOLR','LINM'/
      DATA NTBLS/10/
C
C                   INITIALIZE THE DUMMY ARRAYS
C
C          TBL ASSIGNMENTS FOR SUBROUTINE BLLIND
      DATA TBL1/'KGS ','KGD ','KGI ','FG  ','KE  ','FE  ','RES ','DLE ',
     * 'EB  ','PB  '/
C
C          TBL ASSIGNMENTS FOR SUBROUTINE BLNLIN
      DATA TBL2/'KGS ','KGD ','KGI ','FG  ','KE  ','FE  ','RES ','DLE ',
     * 'DLG ','ME  '/
C
C          TBL ASSIGNMENTS FOR SUBROUTINE BLTEMP
      DATA TBL3/'KGS ','KGD ','KGI ','FG  ','KE  ','FE  ','RES ',
     * 'DLE ','DLG ','ME  ','DLEO','DLGO','FGO '/
C
C          TBL ASSIGNMENTS FOR SUBROUTINE BLVALP
      DATA TBL4/'KGS ','KGD ','MGS ','MGD ','FG  ','KE  ','FE  ',
     * 'DLE ','RES ','DLG ','P   ','LAMB','LAM1','R   ','PHI ',
     * 'KSS ','MSS ','V1  ','VX  ','V2  '/
C
C          TBL ASSIGNMENTS FOR SUBROUTINE BLCOOR
      DATA TBL5/'CORG','DLNC'/
C
C          TBL ASSIGNMENTS FOR SUBROUTINE BLCOND
      DATA TBL6/'NEQ ','DIMP'/
C
C          TBL ASSIGNMENTS FOR SUBROUTINE BLPREL
      DATA TBL7/'PREG','V   '/
C
```

```
C         TBL ASSIGNMENTS FOR SUBROUTINE BLELEM
      DATA TBL8/'LD  ','LDCE','CORE','NE  ','PRNE','PREE'/
C
C         TBL ASSIGNMENTS FOR SUBROUTINE BLSOLR
      DATA TBL9/'FS  ','KE  ','FE  ','DLE ','KSS ','KSD ','KSI ',
     + 'RES '/
C
C         TBL ASSIGNMENTS FOR SUBROUTINE BLLINY
      DATA TBL10/'KSS ','KSD ','KSI ','FS  ','KE  ','FE  ','RES ',
     + 'DLE '/
C
C______ DETERMINE THE CALLING ROUTINE
      DO 5 I = 1,NTBLS
        IF(WHO.EQ.CALLER(I)) IPOINT = I
5     CONTINUE
C
C______ BRANCH TO CORRECT INITIALIZATION LOOP
      GOTO(10,20,30,40,50,60,70,80,90,100),IPOINT
C
C______ INITIALIZE THE REAL ARRAY FOR BLLIND
10    DO 15 I = 1,10
        TBL(I) = TBL1(I)
15    CONTINUE
      RETURN
C
C______ INITIALIZE THE REAL ARRAY FOR BLNLIN
20    DO 25 I = 1,10
        TBL(I) = TBL2(I)
25    CONTINUE
      RETURN
C
C______ INITIALIZE THE REAL ARRAY FOR BLTEMP
30    DO 35 I = 1,13
        TBL(I) = TBL3(I)
35    CONTINUE
      RETURN
C
C______ INITIALIZE THE REAL ARRAY FOR BLVALP
40    DO 45 I = 1,20
        TBL(I) = TBL4(I)
45    CONTINUE
      RETURN
C
C______ INITIALIZE THE REAL ARRAY FOR BLCOOR
50    DO 55 I = 1,2
        TBL(I) = TBL5(I)
55    CONTINUE
      RETURN
C
C______ INITIALIZE THE REAL ARRAY FOR BLCOND
60    DO 65 I = 1,2
```

```
      TBL(I) = TBL6(I)
65    CONTINUE
      RETURN
C
C______ INITIALIZE THE REAL ARRAY FOR BLPREL
70    DO 75 I = 1,2
        TBL(I) = TBL7(I)
75    CONTINUE
      RETURN
C
C______ INITIALIZE THE REAL ARRAY FOR BLELEM
80    DO 85 I = 1,6
        TBL(I) = TBL8(I)
85    CONTINUE
      RETURN
C
C______ INITIALIZE THE REAL ARRAY FOR BLSOLR
90    DO 95 I = 1,8
        TBL(I) = TBL9(I)
95    CONTINUE
      RETURN
C
C______ INITIALIZE THE REAL ARRAY FOR BLLINM
100   DO 105 I = 1,8
        TBL(I) = TBL10(I)
105   CONTINUE
      RETURN
      END


      SUBROUTINE INITPG(IPGKED,WHO)
C=====================================================================
C       THIS SUBROUTINE EXISTS SOLELY TO GET AROUND A MICROSOFT
C       COMPILER BUG.  ITS PURPOSE IS TO INITIALIZE THE ARRAY
C       PASSED AS AN ARGUMENT.  THE DUMMY ARRAY IPGKX HAS BEEN
C       GIVEN THE ATTRIBUTE $NOTLARGE, AND WILL BE INITIALIZED
C       PROPERLY BY THE COMPILER.  THE $NOTLARGE ATTRIBUTE IS
C       ASSIGNED BY DEFAULT SINCE THE DIMENSION DOES NOT EXCEED
C       64K BYTES OF STORAGE.
C
C       THIS SUBROUTINE IS CALLED BY SUBROUTINES ELEM01, ELEM02,
C       AND ELEM07.
C=====================================================================
      DIMENSION IPGKED(*),IPGK1(3),IPGK2(2),IPGK7(3)
      CHARACTER*4 WHO,CALLER(3)
      DATA CALLER/'EL01','EL02','EL07'/
      DATA NCLRS/3/
C
C             HERE IS THE INITIALIZATION FOR ELEM01

      DATA IPGK1/3,3,3/
```

```
C
C           HERE IS THE INITIALIZATION FOR ELEM02
      DATA IPGK2/3,3/
C
C           HERE IS THE INITIALIZATION FOR ELEM07
      DATA IPGK7/2,2,2/
C
C______ DETERMINE THE CALLING ROUTINE
      DO 5 I = 1,NCLRS
        IF(WHO.EQ.CALLER(I)) IPOINT = I
5     CONTINUE
C
C______ BRANCH TO CORRECT INITIALIZATION LOOP
      GOTO(10,20,70),IPOINT
C
C           INITIALIZE IPGKED FOR SUBROUTINE ELEM01
10    DO 15 I = 1,3
        IPGKED(I) = IPGK1(I)
15    CONTINUE
      RETURN
C
C           INITIALIZE IPGKED FOR SUBROUTINE ELEM02
20    DO 25 I = 1,2
        IPGKED(I) = IPGK2(I)
25    CONTINUE
      RETURN
C
C           INITIALIZE IPGKED FOR SUBROUTINE ELEM07
70    DO 75 I = 1,3
        IPGKED(I) = IPGK7(I)
75    CONTINUE
      RETURN
      END
```

```
$LARGE
$NOFLOATCALLS
      SUBROUTINE ELEMLB(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C=======================================================================
C     TO COMPUTE ELEMENT INFORMATIONS FOR ALL TYPES OF ELEMENTS
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/RGDT/IEL,ITPE,NULL(13)
      DIMENSION VCORE(*),VPRNE(*),VPREE(*),VDLE(*),VKE(*),VFE(*)
C-----------------------------------------------------------------------
      GO TO ( 10, 20, 30, 40, 50, 60, 70, 80, 90,100),ITPE
C------- ELEMENT OF TYPE 1
10    CALL ELEM01(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
      GO TO 900
C------- ELEMENT OF TYPE 2
20    CALL ELEM02(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
      GO TO 900
C------- ELEMENT OF TYPE 3
30    CALL ELEM03(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
      GO TO 900
C------- ELEMENT OF TYPE 4
40    CALL ELEM04(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
      GO TO 900
C------- ELEMENT OF TYPE 5
50    CALL ELEM05(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
      GO TO 900
C------- ELEMENT OF TYPE 6
60    CALL ELEM06(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
      GO TO 900
C------- ELEMENT OF TYPE 7
70    CALL ELEM07(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
      GO TO 900
C------- ELEMENT OF TYPE 8
80    CALL ELEM08(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
      GO TO 900
C------- ELEMENT OF TYPE 9
90    CALL ELEM09(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
      GO TO 900
C------- ELEMENT OF TYPE 10
100   CALL ELEM10(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
      GO TO 900
C------- OTHER ELEMENTS
C     ........
900   RETURN
      END
```

```
$LARGE
$NOFLOATCALLS
      SUBROUTINE ELEM01(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C=====================================================================
C     QUADRATIC ELEMENT FOR ANISOTROPIC HARMONIC PROBLEMS
C       IN 1,2 OR 3 DIMENSIONS :
C         1 DIMENSION:  3 NODES ELEMENT
C         2 DIMENSIONS: 8 NODES ISOPARAMETRIC ELEMENT
C         3 DIMENSIONS: 20 NODES ISOPARAMETRIC ELEMENT
C     NUMBER OF INTEGRATION POINTS : 2 IN EACH DIRECTION
C     NUMBER OF DEGREES OF FREEDOM PER NODE : 1
C     ELEMENT MATRIX OR VECTOR FORMED BY THIS SUBPROGRAM
C       ACCORDING TO ICODE VALUE :
C          ICODE.EQ.1  RETURN OF PARAMETERS
C          ICODE.EQ.2  EVALUATE INTERPOLATION FUNCTIONS AND
C                      NUMERICAL INTEGRATION COEFFICIENTS
C          ICODE.EQ.3  ELEMENT MATRIX (VKE)
C          ICODE.EQ.4  TANGENT MATRIX (VKE)....NOT WRITTEN....
C          ICODE.EQ.5  MASS MATRIX (VKE)
C          ICODE.EQ.6  K.U PRODUCT (VFE)
C          ICODE.EQ.7  ELEMENT LOAD (VFE)....NOT WRITTEN....
C          ICODE.EQ.8  PRINT GRADIENTS
C     ELEMENT PROPERTIES
C          VPREE(1)  COEFFICIENT DX
C          VPREE(2)  COEFFICIENT DY
C          VPREE(3)  COEFFICIENT DZ
C        . VPREE(4)  SPECIFIC HEAT CAPACITY C
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNULL(3),FNULL(3)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,IPG
     1  ,ICODE,IDLE0,INEL0,IPG0
      COMMON/ES/M,MR,MP,MDUMMY(10)
      DIMENSION VCORE(*),VPRNE(*),VPREE(*),VDLE(*),VKE(*),VFE(*)
C.......  CHARACTERISTIC DIMENSIONS OF THE ELEMENT
C           (VALID UP TO 3 DIMENSIONS)
C     DIMENSION VCPG(IPG),VKPG(NDIM*IPG),XYZ(NDIM)
      DIMENSION VCPG( 9),VKPG(      27),XYZ(   3)
C     DIMENSION VJ (NDIM*NDIM),VJ1(NDIM*NDIM)
      DIMENSION VJ (        9),VJ1(        9)
C     DIMENSION VNIX( INEL*NDIM),VNI ((1+NDIM)*INEL*IPG),IPGKED(NDIM)
      DIMENSION VNIX(        60),VNI (               2160),IPGKED(   3)
      DATA ZERO/0.D0/,EPS/1.D-6/
C
C              NUMBER OF G.P. IN KSI,ETA,DZETA DIRECTION
C
C+++       THIS IS COMMENTED OUT BECAUSE OF THE MS FORTRAN COMP-
C+++       ILER BUG WHICH WILL NOT INITIALIZE $LARGE ARRAYS.
C+++       THIS ARRAY IS NOW INITIALIZED BY A CALL TO A DUMMY
C+++       SUBROUTINE INITPG WHICH EXISTS SOLELY TO INITIALIZE
```

```
C+++       THE NUMBER OF GAUSS POINTS FOR THE CALLING ROUTINE.
C
C     DATA IPGKED/3,3,3/
C
C         HERE IS THE CALL TO GET AROUND THE COMPILER BUG
C
      CALL INITPG(IPGKED,'EL01')
C
C+++      ALL OF THIS WAS SOLELY TO GET AROUND THE MICROSOFT
C+++      COMPILER BUG
C.......
      IKE=IDLE*(IDLE+1)/2
C
C------  CHOOSE FUNCTION TO BE EXECUTED
C
      GO TO (100,200,300,400,500,600,700,800),ICODE
C
C------  RETURN ELEMENT PARAMETERS IN COMMON 'RGDT'
C
100   GO TO (110,120,130),NDIM
110   IDLEO=3
      INELO=3
      IPGO=3
      RETURN
120   IDLEO=8
      INELO=8
      IPGO=9
      RETURN
130   IDLEO=20
      INELO=20
      IPGO=27
      RETURN
C
C------  EVALUATE COORDINATES, WEIGHTS, FUNCTIONS N AND
C------    THEIR DERIVATIVES AT G.P.
C
200   CALL GAUSS(IPGKED,NDIM,VKPG,VCPG,IPG)
      CALL NI01(VKPG,VNI)
      RETURN
C
C------  COMPUTE ELEMENT STIFFNESS MATRIX
C
C------  INITIALIZE VKE
300   DO 310 I=1,IKE
310   VKE(I)=ZERO
C------  LOOP OVER THE INTEGRATION POINTS
      INI=1+INEL
      DO 330 IG=1,IPG
C------  EVALUATE THE JACOBIAN MATRIX, ITS INVERSE AND ITS DETERMINANT
      CALL JACOB(VNI(INI),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
      IF(DETJ.LT.EPS) WRITE(MP,2000) IEL,IG,DETJ
```

```
2000  FORMAT(' *** ELEM ',I5,' P.G. ',I3,' DET(J)=',E12.5)
C------   PERFORM DETJ*WEIGHT
      COEF=VCPG(IG)*DETJ
C------   EVALUATE FUNCTIONS D(NI)/D(X)
      CALL DNIDX(VNI(INI),VJ1,NDIM,INEL,VNIX)
C------   ACCUMULATE TERMS OF THE ELEMENT MATRIX
      IK=0
      DO 320 J=1,IDLE
      DO 320 I=1,J
      I1=I
      I2=J
      C=ZERO
      DO 315 IJ=1,NDIM
      C=C+VNIX(I1)*VNIX(I2)*VPREE(IJ)
      I1=I1+IDLE
315   I2=I2+IDLE
      IK=IK+1
320    VKE(IK)=VKE(IK)+C*COEF
C------   NEXT G.P.
330   INI=INI+(NDIM+1)*INEL
      RETURN
C
C------   EVALUATE ELEMENT TANGENT MATRIX
C
400   CONTINUE
      RETURN
C
C------   MASS MATRIX
C
500   DO 510 I=1,IKE
510   VKE(I)=ZERO
      IF(VPREE(4).EQ.ZERO)RETURN
      INI=0
      DO 530 IG=1,IPG
C------   EVALUATE THE JACOBIAN MATRIX
      I1=INI+INEL+1
      CALL JACOB(VNI(I1),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
C------   COMPUTE THE WEIGHT
      COEF=VCPG(IG)*DETJ*VPREE(4)
C------   TERMS OF THE MASS MATRIX
      IK=0
      DO 520 J=1,IDLE
      DO 520 I=1,J
      IK=IK+1
      I1=INI+I
      I2=INI+J
520   VKE(IK)=VKE(IK)+VNI(I1)*VNI(I2)*COEF
530   INI=INI+(NDIM+1)*INEL
      RETURN
C
C------   EVALUATE THE ELEMENT RESIDUAL
```

```
C
600   DO 605 I=1,INEL
605   VFE(I)=ZERO
      INI=1+INEL
      DO 640 IG=1,IPG
C------ EVALUATE THE JACOBIAN MATRIX AND THE DERIVATIVES OF N IN X,Y,Z
      CALL JACOB(VNI(INI),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
      CALL DNIDX(VNI(INI),VJ1,NDIM,INEL,VNIX)
C------ COMPUTE THE COMMON COEFFICIENT
      COEF=VCPG(IG)*DETJ
C------ VPREE*B*VDLE PRODUCT
      I1=0
      DO 620 I=1,NDIM
      C=ZERO
      DO 610 J=1,INEL
      I1=I1+1
610   C=C+VNIX(I1)*VDLE(J)
620   VJ(I)=C*COEF*VPREE(I)
C------ (BT)*VJ PRODUCT
      DO 630 I=1,INEL
      I1=I-INEL
      DO 630 J=1,NDIM
      I1=I1+INEL
630   VFE(I)=VFE(I)+VNIX(I1)*VJ(J)
640   INI=INI+(NDIM+1)*INEL
      RETURN
C
C------ EVALUATE FE
C
700   CONTINUE
      RETURN
C
C------ EVALUATE AND PRINT GRADIENTS AT G.P.
C
800   WRITE(MP,2010) IEL
2010  FORMAT(//' GRADIENTS IN ELEMENT :',I4//)
      IDECL=(NDIM+1)*INEL
      INI0=1
      INI=1+INEL
      DO 830 IG=1,IPG
      CALL JACOB(VNI(INI),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
      CALL DNIDX(VNI(INI),VJ1,NDIM,INEL,VNIX)
C------ EVALUATE THE COORDINATES OF THE G.P.
      DO 803 I=1,NDIM
803   XYZ(I)=ZERO
      IC=1
      I0=INI0
      DO 807 IN=1,INEL
      C=VNI(I0)
      DO 805 I=1,NDIM
      XYZ(I)=XYZ(I)+C*VCORE(IC)
```

```
805   IC=IC+1
807   IO=IO+1
C------  EVALUATE THE GRADIENT
      I1=0
      DO 820 I=1,NDIM
      C=ZERO
      DO 810 J=1,IDLE
      I1=I1+1
810   C=C+VNIX(I1)*VDLE(J)
820   VJ(I)=C*VPREE(I)
C------  PRINT THE GRADIENT
      WRITE(MP,2020) IG,(XYZ(I),I=1,NDIM)
2020  FORMAT(5X,'P.G. :',I3,' COORDINATES :',3E12.5)
      WRITE(MP,2025)(VJ(I),I=1,NDIM)
2025  FORMAT(15X,'GRADIENTS   :',3E12.5)
      INIO=INIO+IDECL
830   INI=INI+IDECL
      WRITE(MP,2030)
2030  FORMAT(//)
      RETURN
      END


      SUBROUTINE NIO1(VKPG,VNI)
C=====================================================================
C     TO EVALUATE THE INTERPOLATION FUNCTIONS AND THEIR DERIVATIVES
C     D(N)/D(KSI) D(N)/D(ETA) BY THE GENERAL PN-INVERSE METHOD
C     FOR 1,2 OR 3 DIMENSIONAL QUADRATIC ELEMENTS
C       INPUT
C          VKPG    COORDINATES AT WHICH N IS TO BE EVALUATED
C          IPG     NUMBER OF POINTS
C          INEL    NUMBER OF FUNCTIONS N (OR OF NODES)      INEL.LE.20
C          NDIM    NUMBER OF DIMENSIONS                     NDIM.LE.3
C       OUTPUT
C          VNI     FUNCTIONS N AND DERIVATIVES
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNULL(3),FNULL(3)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,IPG
     1  ,NULL(4)
      COMMON/TRVL/VKSI,VPN,VP,KEXP,KDER,K1
      DIMENSION VKPG(*),VNI(*)
      DIMENSION VKSI1(3),KEXP1(3),VKSI2(16),KEXP2(16),VKSI3(60),
     1  KEXP3(60)
C
C.......  INFORMATION TO DEFINE THE 3 REFERENCE ELEMENTS
C          (INEL.LE.20  NDIM.LE.3)
C     DIMENSION VKSI(NDIM*INEL),KEXP(NDIM*INEL),KDER(NDIM)
      DIMENSION VKSI(       60),KEXP(       60),KDER(   3)
C     DIMENSION VPN (INEL*INEL),VP(INEL)
      DIMENSION VPN (       400),VP(   20)
```

```
C     DIMENSION K1(INEL)
      DIMENSION K1(   20)
C        CHARACTERISTICS FOR 1,2 AND 3 DIMENSIONAL REFERENCE ELEMENTS
      DATA IDEGR/2/
C
C+++      THIS IS COMMENTED OUT BECAUSE OF THE MS FORTRAN COMP-
C+++      ILER BUG WHICH WILL NOT INITIALIZE $LARGE ARRAYS.
C+++      THESE ARRAYS ARE NOW INITIALIZED BY A CALL TO A DUMMY
C+++      SUBROUTINE INITN1 WHICH EXISTS SOLELY TO INITIALIZE
C+++      THESE SIX ARRAYS.
C
C     DATA VKSI1/-1.D0,0.D0,1.D0/,KEXP1/0,1,2/
C     DATA VKSI2/-1.D0,-1.D0, +0.D0,-1.D0, +1.D0,-1.D0, +1.D0,+0.D0,
C    1           +1.D0,+1.D0, +0.D0,+1.D0, -1.D0,+1.D0, -1.D0,+0.D0/
C     DATA KEXP2/0,0, 1,0, 0,1, 2,0, 1,1, 0,2, 2,1, 1,2/
C     DATA VKSI3/-1.D0,-1.D0,-1.D0, +0.D0,-1.D0,-1.D0,
C    1           +1.D0,-1.D0,-1.D0, +1.D0,+0.D0,-1.D0,
C    2           +1.D0,+1.D0,-1.D0, +0.D0,+1.D0,-1.D0,
C    3           -1.D0,+1.D0,-1.D0, -1.D0,+0.D0,-1.D0,
C    4           -1.D0,-1.D0,+0.D0, +1.D0,-1.D0,+0.D0,
C    5           +1.D0,+1.D0,+0.D0, -1.D0,+1.D0,+0.D0,
C    6           -1.D0,-1.D0,+1.D0, +0.D0,-1.D0,+1.D0,
C    7           +1.D0,-1.D0,+1.D0, +1.D0,+0.D0,+1.D0,
C    8           +1.D0,+1.D0,+1.D0, +0.D0,+1.D0,+1.D0,
C    9           -1.D0,+1.D0,+1.D0, -1.D0,+0.D0,+1.D0/
C     DATA KEXP3/0,0,0, 1,0,0, 0,1,0, 0,0,1, 1,1,1,
C    1  1,1,0, 0,1,1, 1,0,1, 2,0,0, 0,2,0, 0,0,2,
C    2  2,1,0, 2,0,1, 2,1,1, 1,2,0, 0,2,1, 1,2,1,
C    3  1,0,2, 0,1,2, 1,1,2/
C
C         HERE IS THE CALL TO GET AROUND THE MICROSOFT
C         COMPILER BUG
C
      CALL INITN1(VKSI1,KEXP1,VKSI2,KEXP2,VKSI3,KEXP3)
C
C+++      ALL OF THIS WAS SIMPLY TO GET AROUND THE
C+++      MICROSOFT COMPILER BUG
C
C.......
      IDEG=IDEGR
C------- SELECT TABLES VKSI AND KEXP ACCORDING TO NDIM
      I1=NDIM*INEL
      DO 5 I=1,I1
      GO TO (1,2,3),NDIM
1     VKSI(I)=VKSI1(I)
      KEXP(I)=KEXP1(I)
      GO TO 5
2     VKSI(I)=VKSI2(I)
      KEXP(I)=KEXP2(I)
      GO TO 5
3     VKSI(I)=VKSI3(I)
```

```
      KEXP(I)=KEXP3(I)
5     CONTINUE
C------   EVALUATE THE PN-INVERSE MATRIX
      CALL PNINV(VKSI,KEXP,VP,K1,VPN)
C------   EVALUATE N,D(N)/D(KSI),D(N)/D(ETA) AT G.P.
      I1=1
      I2=1
      DO 10 IG=1,IPG
      KDER(1)=0
      KDER(2)=0
      KDER(3)=0
      CALL NI(VKPG(I1),KEXP,KDER,VP,VPN,VNI(I2))
      I2=I2+INEL
      KDER(1)=1
      CALL NI(VKPG(I1),KEXP,KDER,VP,VPN,VNI(I2))
      I2=I2+INEL
      IF(NDIM.EQ.1) GO TO 10
      KDER(1)=0
      KDER(2)=1
      CALL NI(VKPG(I1),KEXP,KDER,VP,VPN,VNI(I2))
      I2=I2+INEL
      IF(NDIM.EQ.2) GO TO 10
      KDER(2)=0
      KDER(3)=1
      CALL NI(VKPG(I1),KEXP,KDER,VP,VPN,VNI(I2))
      I2=I2+INEL
10    I1=I1+NDIM
      RETURN
      END
```

```
      SUBROUTINE ELEM02(VCORE,VPRNE,VPREE,VDLE,VKE,VFE)
C======================================================================
C     8 NODES QUADRATIC ELEMENT FOR 2 DIMENSIONAL ELASTICITY
C          EVALUATE ELEMENT INFORMATIONS ACCORDING TO ICODE VALUE
C          ICODE=1  ELEMENT PARAMETERS
C          ICODE=2  INTERPOLATION FUNCTIONS AND GAUSS COEFFICIENTS
C          ICODE=3  STIFFNESS MATRIX
C          ICODE=4  TANGENT MATRIX  ... NOT WRITTEN ...
C          ICODE=5  MASS MATRIX
C          ICODE=6  RESIDUALS
C          ICODE=7  SECOND MEMBER
C          ICODE=8  EVALUATE AND PRINT STRESSES
C       ELEMENT PROPERTIES
C          VPREE(1)  YOUNG'S MODULUS
C          VPREE(2)  POISSON'S COEFFICIENT
C          VPREE(3)  .EQ.0  PLANE STRESS
C                    .EQ.1  PLANE STRAIN
C          VPREE(4)  SPECIFIC MASS
C======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNULL(3),FNULL(3)
      COMMON/ASSE/NSYM,MFILLR(3)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,IPG
     1  ,ICODE,IDLE0,INEL0,IPG0
      COMMON/ES/M,MR,MP,MDUMMY(10)
      DIMENSION VCORE(*),VPRNE(*),VPREE(*),VDLE(*),VKE(*),VFE(*)
C....... CHARACTERISTIC DIMENSIONS OF THE ELEMENT
C     DIMENSION VCPG(      IPG),VKPG(NDIM*IPG),VDE1(IMATD**2)
      DIMENSION VCPG(         9),VKPG(      18),VDE1(        9)
C     DIMENSION VBE (IMATD*IDLE),VDE (IMATD**2),VJ (NDIM*NDIM),VJ1(NDIM*
      DIMENSION VBE (        48),VDE (        9),VJ (         4),VJ1(4)
C     DIMENSION VNIX( INEL*NDIM),VNI ((1+NDIM)*INEL*IPG),IPGKED(NDIM)
      DIMENSION VNIX(        16),VNI (                216),IPGKED(   2)
C.......
      DATA ZERO/0.D0/,DEUX/2.D0/,X05/0.5D0/,RADN/.572957795130823D2/
      DATA EPS/1.D-6/
      SQRT(X)=DSQRT(X)
      ATAN2(X,Y)=DATAN2(X,Y)
C        DIMENSION OF MATRIX D
      DATA IMATD/3/
C
C+++       THIS IS COMMENTED OUT BECAUSE OF THE MS FORTRAN COMP-
C+++       ILER BUG WHICH WILL NOT INITIALIZE $LARGE ARRAYS.
C+++       THIS ARRAY IS NOW INITIALIZED BY A CALL TO A DUMMY
C+++       SUBROUTINE INITPG WHICH EXISTS SOLELY TO INITIALIZE
C+++       THE NUMBER OF GAUSS POINTS FOR THE CALLING ROUTINE.
C
C     DATA IPGKED/3,3/
C
C        HERE IS THE CALL TO GET AROUND THE COMPILER BUG
```

```
C
      CALL INITPG(IPGKED,'EL02')
C
C+++       ALL OF THIS WAS SOLELY TO GET AROUND THE MICROSOFT
C+++       COMPILER BUG
C
C-------  CHOOSE FUNCTION TO BE EXECUTED
C
      GO TO (100,200,300,400,500,600,700,800),ICODE
C
C-------  RETURN ELEMENT PARAMETERS IN COMMON 'RGDT'
C
100   IDLE0=16
      INEL0=8
      IPG0=9
C     RETURN
C
C-------  EVALUATE COORDINATES, WEIGHTS, FUNCTIONS N AND THEIR
C-------  DERIVATIVES AT G.P.
C
200   CALL GAUSS(IPGKED,NDIM,VKPG,VCPG,IPG)
      IF(M.LT.2) GO TO 220
      WRITE(MP,2000) IPG
2000  FORMAT(/I5,'  GAUSS POINTS'/10X,'VCPG',25X,'VKPG')
      I0=1
      DO 210 IG=1,IPG
      I1=I0+NDIM-1
      WRITE(MP,2010) VCPG(IG),(VKPG(I),I=I0,I1)
210   I0=I0+NDIM
2010  FORMAT(1X,F20.15,5X,3F20.15)
220   CALL NI02(VKPG,VNI)
      IF(M.LT.2) RETURN
      I1=3*INEL*IPG
      WRITE(MP,2020) (VNI(I),I=1,I1)
2020  FORMAT(/' FUNCTIONS N AND DERIVATIVES'/ (1X,8E12.5))
      RETURN
C
C-------  EVALUATE ELEMENT STIFFNESS MATRIX
C
C-------  INITIALIZE VKE
300   DO 310 I=1,136
310   VKE(I)=ZERO
C-------  FORM MATRIX D
      CALL D02(VPREE,VDE)
      IF(M.GE.2) WRITE(MP,2030) (VDE(I),I=1,9)
2030  FORMAT(/' MATRIX D'/1X,9E12.5)
C-------  LOOP OVER THE G.P.
      I1=1+INEL
      DO 330 IG=1,IPG
C-------  EVALUATE THE JACOBIAN, ITS INVERSE AND ITS DETERMINANT
      CALL JACOB(VNI(I1),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
```

```
      IF(DETJ.LT.EPS) WRITE(MP,2040) IEL,IG,DETJ
2040  FORMAT(' *** ELEM ',I5,' G.P. ',I3,' DET(J)=',E12.5)
      IF(M.GE.2) WRITE(MP,2050) VJ,VJ1,DETJ
2050  FORMAT(/' JACOBIAN=',4E12.5 / ' J INVERS=',4E12.5/' DETJ=',E12.5)
C-------  PERFORM D*COEF
      C=VCPG(IG)*DETJ
      DO 320 I=1,9
320   VDE1(I)=VDE(I)*C
C-------  FORM MATRIX B
      CALL DNIDX(VNI(I1),VJ1,NDIM,INEL,VNIX)
      IF(M.GE.2) WRITE(MP,2060) (VNIX(I),I=1,16)
2060  FORMAT(/' VNIX'/(1X,8E12.5))
      CALL B02(VNIX,INEL,VBE)
      IF(M.GE.2) WRITE(MP,2070) (VBE(I),I=1,48)
2070  FORMAT(/' MATRIX B'/(1X,10E12.5))
      CALL BTDB(VKE,VBE,VDE1,IDLE,IMATD,NSYM)
330   I1=I1+3*INEL
      RETURN
C
C-------  EVALUATE THE ELEMENT TANGENT MATRIX
C
400   CONTINUE
      RETURN
C
C-------  EVALUATE THE MASS MATRIX
C
500   DO 510 I=1,136
510   VKE(I)=ZERO
C-------  LOOP OVER THE G.P.
      IDIM1=NDIM-1
      IDECL=(NDIM+1)*INEL
      I1=1+INEL
      I2=0
      DO 550 IG=1,IPG
      CALL JACOB(VNI(I1),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
      D=VCPG(IG)*DETJ*VPREE(4)
C-------  ACCUMULATE MASS TERMS
      IDL=0
      DO 540 J=1,INEL
      JJ=I2+J
      J0=1+IDL*(IDL+1)/2
      DO 530 I=1,J
      II=I2+I
      C=VNI(II)*VNI(JJ)*D
      VKE(J0)=VKE(J0)+C
      IF(NDIM.EQ.1) GO TO 530
      J1=J0+IDL+2
      DO 520 II=1,IDIM1
      VKE(J1)=VKE(J1)+C
520   J1=J1+J1+1
530   J0=J0+NDIM
```

```
540   IDL=IDL+NDIM
      I1=I1+IDECL
550   I2=I2+IDECL
      RETURN
C
C-------  EVALUATE THE ELEMENT RESIDUAL
C
C-------  FORM MATRIX D
600   CALL D02(VPREE,VDE)
C-------  INITIALIZE THE RESIDUAL VECTOR
      DO 610 ID=1,IDLE
610   VFE(ID)=ZERO
C-------  LOOP OVER THE G.P.
      I1=1+INEL
      DO 640 IG=1,IPG
C-------  EVALUATE THE JACOBIAN
      CALL JACOB(VNI(I1),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
C-------  EVALUATE FUNCTIONS D(NI)/D(X)
      CALL DNIDX(VNI(I1),VJ1,NDIM,INEL,VNIX)
C-------  EVALUATE STRAINS AND STRESSES
      EPSX=ZERO
      EPSY=ZERO
      GAMXY=ZERO
      ID=1
      DO 620 IN=1,INEL
      UN=VDLE(ID)
      VN=VDLE(ID+1)
      C1=VNIX(IN)
      IN1=IN+INEL
      C2=VNIX(IN1)
      EPSX=EPSX+C1*UN
      EPSY=EPSY+C2*VN
      GAMXY=GAMXY+C1*VN+C2*UN
620   ID=ID+2
      C1=VCPG(IG)*DETJ
      C2=VDE(2)*C1
      C3=VDE(9)*C1
      C1=VDE(1)*C1
      SIGX=C1*EPSX+C2*EPSY
      SIGY=C2*EPSX+C1*EPSY
      TAUXY=C3*GAMXY
C-------  FORM THE RESIDUAL
      ID=1
      DO 630 IN=1,INEL
      C1=VNIX(IN)
      IN1=IN+INEL
      C2=VNIX(IN1)
      VFE(ID)=VFE(ID)+C1*SIGX+C2*TAUXY
      VFE(ID+1)=VFE(ID+1)+C2*SIGY+C1*TAUXY
630   ID=ID+2
640   I1=I1+3*INEL
```

```
      RETURN
C
C------- EVALUATE VOLUMIC FORCES, FX,FY PER UNIT VOLUME
C         ( FOR GRAVITY FX=0 FY=-VPREE(4) )
C
700   FX=ZERO
      FY=-VPREE(4)
      DO 710 I=1,16
710   VFE(I)=ZERO
      I1=1
      IDECL=(NDIM+1)*INEL
      DO 730 IG=1,IPG
      CALL JACOB(VNI(I1+INEL),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
      DX=VCPG(IG)*DETJ
      DY=DX*FY
      DX=DX*FX
      I2=I1
      I3=1
      DO 720 IN=1,INEL
      VFE(I3)=VFE(I3)+DX*VNI(I2)
      VFE(I3+1)=VFE(I3+1)+DY*VNI(I2)
      I2=I2+1
720   I3=I3+2
730   I1=I1+IDECL
      RETURN
C
C------- EVALUATE AND PRINT STRESSES AT G.P.
C
800   WRITE(MP,2080) IEL
2080  FORMAT(//' STRESSES IN ELEMENT ',I5/
     1 '  P.G.',7X,'X',11X,'Y',9X,'EPSX',8X,'EPSY',7X,'GAMXY',8X,'SIGX',
     2 8X,'SIGY',7X,'TAUXY',8X,'TETA'/ 71X ,'SIG1',8X,'SIG2',7X,'TAUMAX'
     3 /)
C------- FORM THE MATRIX D
      CALL D02(VPREE,VDE)
C------- LOOP OVER THE G.P.
      I1=1+INEL
      I2=0
      DO 820 IG=1,IPG
C------- EVALUATE THE JACOBIAN
      CALL JACOB(VNI(I1),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
C------- EVALUATE FUNCTIONS D(NI)/D(X)
      CALL DNIDX(VNI(I1),VJ1,NDIM,INEL,VNIX)
C------- COMPUTE STRAINS AND COORDINATES AT G.P.
      EPSX=ZERO
      EPSY=ZERO
      GAMXY=ZERO
      X=ZERO
      Y=ZERO
      ID=1
      DO 810 IN=1,INEL
```

```
      UN=VDLE(ID)
      VN=VDLE(ID+1)
      XN=VCORE(ID)
      YN=VCORE(ID+1)
      C1=VNIX(IN)
      IN1=IN+INEL
      C2=VNIX(IN1)
      IN1=IN+I2
      C3=VNI(IN1)
      EPSX=EPSX+C1*UN
      EPSY=EPSY+C2*VN
      GAMXY=GAMXY+C1*VN+C2*UN
      X=X+C3*XN
      Y=Y+C3*YN
810   ID=ID+2
C------- COMPUTE THE STRESSES
      SIGX=VDE(1)*EPSX+VDE(2)*EPSY
      SIGY=VDE(2)*EPSX+VDE(1)*EPSY
      TAUXY=VDE(9)*GAMXY
C------- COMPUTE THE PRINCIPAL STRESSES
      TETA=ATAN2(DEUX*TAUXY,SIGX-SIGY)*X05
      TETA=TETA*RADN
      C1=(SIGX+SIGY)*X05
      C2=(SIGX-SIGY)*X05
      TAUMAX=SQRT(C2*C2+TAUXY*TAUXY)
      SIG1=C1+TAUMAX
      SIG2=C1-TAUMAX
      WRITE(MP,2090) IG,X,Y,EPSX,EPSY,GAMXY,SIGX,SIGY,TAUXY,
     1  TETA,SIG1,SIG2,TAUMAX
2090  FORMAT(1X,I5,8E12.5,5X,F5.1/66X,3E12.5)
      I2=I2+3*INEL
820   I1=I1+3*INEL
      RETURN
      END
```

```
      SUBROUTINE NI02(VKPG,VNI)
C=================================================================
C     TO EVALUATE THE INTERPOLATION FUNCTIONS N AND THEIR DERIVATIVES
C     D(N)/D(KSI) AND D(N)/D(ETA) BY GENERAL PN-INVERSE METHOD
C       INPUT
C          VKPG    COORDINATES AT WHICH N IS TO BE EVALUATED
C          IPG     NUMBER OF POINTS
C          INEL    NUMBER OF FUNCTIONS N (OR OF NODES)      INEL.EQ.8
C          NDIM    NUMBER OF DIMENSIONS                     NDIM.EQ.2
C       OUTPUT
C          VNI      FUNCTIONS N AND DERIVATIVES
C=================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/COOR/NDIM,NNULL(3),FNULL(3)
      COMMON/RGDT/IEL,ITPE,ITPE1,IGRE,IDLE,ICE,IPRNE,IPREE,INEL,IDEG,IPG
     1   ,NULL(4)
      COMMON/TRVL/VKSI,VPN,VP,KEXP,KDER,K1,RNULL(420),INULL
      DIMENSION VKPG(*),VNI(*)
C
C.......  INFORMATIONS RELATED TO THE 8 NODES REFERENCE SQUARE ELEMENT
C          (INEL.EQ.8  NDIM.EQ.2)
C     DIMENSION VKSI(NDIM*INEL),KEXP(NDIM*INEL),KDER(NDIM)
      DIMENSION VKSI(       16),KEXP(       16),KDER(   2)
C     DIMENSION VPN (INEL*INEL),VP(INEL),K1(INEL)
      DIMENSION VPN (       64),VP(   8),K1(   8)
C
C         NODAL COORDINATES OF THE REFERENCE ELEMENT
C
      DATA IDEGR/2/
C
C+++       THIS IS COMMENTED OUT BECAUSE OF THE MS FORTRAN COMP-
C+++       ILER BUG WHICH WILL NOT INITIALIZE $LARGE ARRAYS.
C+++       THESE ARRAYS ARE NOW INITIALIZED BY A CALL TO A DUMMY
C+++       SUBROUTINE INITN1 WHICH EXISTS SOLELY TO INITIALIZE
C+++       THESE TWO ARRAYS.
C
C     DATA VKSI/-1.D0,-1.D0, +0.D0,-1.D0, +1.D0,-1.D0, +1.D0,+0.D0,
C    1           +1.D0,+1.D0, +0.D0,+1.D0, -1.D0,+1.D0, -1.D0,+0.D0/
C         MONOMIAL EXPONENTS OF THE POLYNOMIAL BASIS, MAX-DEGREE
C     DATA KEXP/0,0, 1,0, 0,1, 2,0, 1,1, 0,2, 2,1, 1,2/
C
C          HERE IS THE CALL TO GET AROUND THE MICROSOFT
C          COMPILER BUG
C
      CALL INITN2(VKSI,KEXP)
C
C+++       ALL OF THIS HAS BEEN TO GET AROUND THE
C+++       COMPILER BUG.
C.......
      IDEG=IDEGR
```

```
C-------  EVALUATE THE PN-INVERSE MATRIX
      CALL PNINV(VKSI,KEXP,VP,K1,VPN)
C-------  EVALUATE N,D(N)/D(KSI),D(N)/D(ETA) AT G.P.
      I1=1
      I2=1
      DO 10 IG=1,IPG
      KDER(1)=0
      KDER(2)=0
      CALL NI(VKPG(I1),KEXP,KDER,VP,VPN,VNI(I2))
      I2=I2+INEL
      KDER(1)=1
      CALL NI(VKPG(I1),KEXP,KDER,VP,VPN,VNI(I2))
      I2=I2+INEL
      KDER(1)=0
      KDER(2)=1
      CALL NI(VKPG(I1),KEXP,KDER,VP,VPN,VNI(I2))
      I2=I2+INEL
10    I1=I1+NDIM
      RETURN
      END
```

```
      SUBROUTINE DO2(VPREE,VDE)
C=====================================================================
C     TO FORM MATRIX D (2 DIMENSIONAL ELASTICITY)
C        INPUT
C          VPREE   ELEMENT PROPERTIES
C                        VPREE(1)   YOUNG'S MODULUS
C                        VPREE(2)   POISSON'S COEFFICIENT
C                        VPREE(3)  .EQ.0  PLANE STRESSES
C                                  .EQ.1  PLANE STRAINS
C        OUTPUT
C          VDE     MATRIX D (FULL)
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION VPREE(*),VDE(9)
      DATA ZERO/0.D0/,UN/1.D0/,DEUX/2.D0/
      E=VPREE(1)
      X=VPREE(2)
      A=VPREE(3)
      C1=E*(UN-A*X)/((UN+X)*(UN-X-A*X))
      C2=C1*X/(UN-A*X)
      C3=E/(DEUX*(UN+X))
      VDE(1)=C1
      VDE(2)=C2
      VDE(3)=ZERO
      VDE(4)=C2
      VDE(5)=C1
      VDE(6)=ZERO
      VDE(7)=ZERO
      VDE(8)=ZERO
      VDE(9)=C3
      RETURN
      END
```

```
      SUBROUTINE B02(VNIX,INEL,VBE)
C=======================================================================
C     TO FORM MATRIX B (2 DIMENSIONAL ELASTICITY)
C        INPUT
C          VNIX    DERIVATIVES OF INTERPOLATION FUNCTIONS W.R.T. X,Y,Z
C          INEL    NUMBER OF INTERPOLATION FUNCTIONS
C        OUTPUT
C          VBE     MATRIX B
C=======================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION VNIX(INEL,*),VBE(3,*)
      DATA ZERO/0.D0/
      J=1
      DO 10 I=1,INEL
      C1=VNIX(I,1)
      C2=VNIX(I,2)
      VBE(1,J)=C1
      VBE(1,J+1)=ZERO
      VBE(2,J)=ZERO
      VBE(2,J+1)=C2
      VBE(3,J)=C2
      VBE(3,J+1)=C1
10    J=J+2
      RETURN
      END
```

```
      SUBROUTINE BTDB(VKE,VBE,VDE,IDLE,IMATD,NSYM)
C=====================================================================
C     TO ADD THE PRODUCT B(T).D.B  TO  VKE
C        INPUT
C          VKE     ELEMENT MATRIX NON SYMMETRICAL  (NSYM.EQ.1)
C                                     SYMMETRICAL  (NSYM.EQ.0)
C          VBE     MATRIX B
C          VDE     MATRIX D (FULL)
C          IDLE    TOTAL NUMBER OF D.O.F. PER ELEMENT
C          IMATD   DIMENSION OF MATRIX D  (MAX. 6)
C        OUTPUT
C          VKE
C=====================================================================
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION VKE(*),VBE(IMATD,*),VDE(IMATD,*),T(6)
      DATA ZERO/0.D0/
C---------------------------------------------------------------------
      IJ=1
      IMAX=IDLE
      DO 40 J=1,IDLE
      DO 20 I1=1,IMATD
      C=ZERO
      DO 10 J1=1,IMATD
10    C=C+VDE(I1,J1)*VBE(J1,J)
20    T(I1)=C
      IF(NSYM.EQ.0) IMAX=J
      DO 40 I=1,IMAX
      C=ZERO
      DO 30 J1=1,IMATD
30    C=C+VBE(J1,I)*T(J1)
      VKE(IJ)=VKE(IJ)+C
40    IJ=IJ+1
      RETURN
      END
```

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	Department Chairman, Code 69 Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943	1
4.	Professor Gilles Cantin, Code 69Ci Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943	4
5.	LCDR Rehe E. Ruesch Naval Ship Repair Facility Subic Bay, Box 34 FPO, San Francisco, CA 96651	2
6.	Professor Young Shin, Code 69Sg Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943	1
7.	Professor K. J. Bathe Mechanical Engineering Department M.I.T. 77 Massachusetts Avenue Cambridge, Massachusetts 02139	1
8.	Professor Edward L. Wilson Structural Engineering Division Civil Engineering Department University of California (Berkeley) Berkeley, California 94720	1
9.	Dr. Gilbert Touzot U.T.C University de Technologie 60206 Compiegne Cedex France	1

10. Professor Guri Dhatt 1
Centre Technique de l'Informatique
Universite Laval
Quebec, Prov. de Quebec
Canada, G1K7P4

11. Dr. Jean Louis Batoz 1
U.T.C.
Universite de Technologie
60206 Compiegne Cedex
France

12. Professor G. N. Vanderplaats 1
Department of Mechanical and
Environmental Engineering
University of California
Santa Barbara, California 93106

13. CAPT G. M. Lachance, USN 1
Code SEA 56
Naval Sea Systems Command
Naval Sea Systems Command Headquarters
Washington, DC 20362

14. CDR W. L. Marsh, USN 1
Code SEA 50D
Naval Sea Systems Command
Naval Sea Systems Command Headquarters
Washington, DC 20362

15. CDR James Buckingham 1
Code SEA 00C
Naval Sea Systems Command
Naval Sea Systems Command Headquarters
Washington, DC 20362

16. LCDR Lael R. Easterling, USN 1
4243 N. W. 54th
Oklahoma City, Oklahoma 73112

17. LT John H. Preisel, Jr., USN 1
922 Bernard Road
Peekskill, New York 10566

18. LT Joseph L. Paquette, USN 1
19311 142nd Place SE
Renton, Washington 98055

19. Mr. Ray Mallory 1
Code 753
Naval Coastal Systems Center
Panama City, Florida 32407

20. Mr. K. Calvin 1
Code SEA 521
Naval Sea Systems Command
Naval Sea Systems Command Headquarters
Washington, DC 20362

21. Mr. J. Kenworthy 1
Code SEA 5244
Naval Sea Systems Command
Naval Sea Systems Command Headquarters
Washington, DC 20362

22. Mr. R. A. Langworthy 1
Applied Technology Laboratories
U.S. Army Research and Technology Laboratory
Fort Eustis, Virginia 23604

23. Mr. E. M. Lenoe 1
Army Materials & Mechanic Research Center
Arsenal Street
Watertown, Massachusetts 02172

24. Dr. H. B. Osborn 1
6337 N. Pomona
Tucson, Arizona 85704

2?0782

Thesis
R842 Ruesch
c.1 Implementation of a
general finite element
code on an IBM PC compatible microcomputer.